TAMPEREEN TEKNILLINEN YLIOPISTO

HARI NORTUNEN

YHTÄLÖRYHMÄN ROBUSTI RATKAISU DIFFERENTIAALI-YHTÄLÖN AVULLA

Diplomityö

Tarkastaja: Prof. Seppo Pohjolainen
Tarkastaja: TkT Lassi Paunonen
Tarkastajat ja aihe hyväksytty
Luonnontieteiden ja ympäristötekniikan
tiedekunnan tiedekuntaneuvoston
kokouksessa 7.4.2010

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Teknis-luonnontieteellinen koulutusohjelma
HARI NORTUNEN: Yhtälöryhmän robusti ratkaisu differentiaaliyhtälön avulla
Diplomityö, 99 sivua, 4 liitesivua
Joulukuu 2011
Pääaine: Matematiikka
Tarkastajat: Prof. Seppo Pohjolainen, TkT Lassi Paunonen
Avainsanat: lineaarialgebra, differentiaaliyhtälöt, systeemiteoria, säätötekniikka, robustisuus, LU-hajotelma, numeerinen ratkaisu

Diplomityön tavoitteena on muodostaa tietokonelaskennan kannalta mahdollisimman tehokas menetelmä, kun lineaarisia yhtälöryhmiä halutaan ratkaista toistuvasti. Lisäehtona on se, että peräkkäisissä yhtälöryhmissä kerroinmatriisi pysyy lähes samana. Mikäli kerroinmatriisien muutokset ovat hyvin pieniä, myös ratkaisujen muutokset ovat pieniä. Tällöin on toivottavaa, että edellisen yhtälöryhmän ratkaisua pystyy jotenkin hyödyntämään, jotta tarkasteltavalle yhtälöryhmälle ei tarvitse hakea ratkaisua kokonaan uudestaan. Tehokkaaksi menetelmäksi osoittautuu robustiin säätötekniikkaan perustuva stabiili differentiaaliyhtälösysteemi, jonka ratkaisuvektorin yksi osa suppenee kohti lineaarisen yhtälöryhmän ratkaisua. Kun saatu ratkaisu asetetaan seuraavan probleeman alkutilaksi, myös seuraava ratkaisu on helppo laskea, koska yhtälöryhmien ratkaisut ovat hyvin lähellä toisiaan. Itse systeemi voidaan ratkaista millä tahansa differentiaaliyhtälöiden ratkaisumenetelmällä; tässä diplomityössä käytetään numeerisia menetelmiä. Yllä esitettyä lähestymistapaa kutsutaan DY-ratkaisijaksi.

Kun DY-ratkaisijan toimintaperiaate ja parametrit on hahmoteltu, menetelmää voidaan verrata lineaaristen yhtälöryhmien ratkaisussa tyypillisesti käytetyn LU-hajotelman kanssa. Vertailussa havaitaan, että DY-ratkaisija on selvästi nopeampi kuin LU-hajotelma niissä tilanteissa, joissa yhtälöryhmän kerroinmatriisi on leveä, eli matriisilla on enemmän sarakkeita kuin vaakarivejä. Ratkaisevaa on myös se, että peräkkäisissä yhtälöryhmissä kerroinmatriisien ja ratkaisuvektorien muutokset pysyvät pieninä. Riittävän pienillä muutoksilla DY-ratkaisija on LU-hajotelmaa tehokkaampi myös neliömatriisin tapauksessa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Science and Engineering
HARI NORTUNEN: The Robust Solution of Linear Equations Using a Differential Equation
Master of Science Thesis, 99 pages, 4 Appendix pages
December 2011
Major: Mathematics
Examiners: Prof. Seppo Pohjolainen, D.Sc. Lassi Paunonen
Keywords: linear algebra, differential equations, systems theory, control theory, robustness, LU decomposition, numerical solution

The purpose of this Master of Science Thesis is to obtain a computationally effective method for solving systems of linear equations repeatedly. As an additional condition, we demand that the coefficient matrix remains approximately the same in consecutive systems. If the changes of the coefficient matrix are very small, the changes of the solution vector will also be small. In that case, it is desirable to be able to use the solution of the previous system, without having to solve the current system from the very beginning. We find out that a stable system of ordinary differential equations is an effective method. One block of the solution vector converges to the solution of the linear equation. This method is based on robust control theory. As the obtained solution is set as the initial state of the next system, the next solution is also easy to compute because the solutions of the linear systems are close to each other. The ODE system can be solved by using any available methods — in this Master of Science Thesis, we use numerical methods. The principle described above is called an ODE solver.

When we have formulated the mechanics and parameters of the ODE solver, we can compare the method with LU decomposition, a method commonly used for solving systems of linear equations. After comparing the two methods, we notice that the ODE solver is clearly faster than the LU decomposition when the coefficient matrix of the linear system has more columns than rows. It is also crucial that the changes in consecutive coefficient matrices and solution vectors remain small. With sufficiently small changes, the ODE solver is more effective than the LU decomposition even when the coefficient matrix is square.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston Matematiikan laitokselle. Se perustuu osittain kesällä 2007 tekemääni matematiikan erikoistyöhön. Aloitin diplomityön suunnittelun ja kehittämisen itsenäisesti keväällä ja kesällä 2010. Siirryttyäni kaksiportaiseen tutkintorakenteeseen kirjoitin kandidaatintyöni syksyllä 2010, ja keväällä 2011 jatkoin työskentelyä diplomityön parissa. Aloitin kirjoitusprosessin maaliskuussa 2011 osa-aikaisella työsopimuksella, jatkoin kirjoittamista kesällä 2011 kokopäiväisesti ja viimeistelin diplomityön valmiiksi osa-aikaisesti syksyllä 2011.

Haluaisin kiittää professori Seppo Pohjolaista, vanhempaa tutkijaa Timo Hämäläistä, tekniikan tohtori Lassi Paunosta sekä tutkija Petteri Laakkosta diplomityön ohjauksesta ja hyvistä neuvoista. Erityiskiitokset koko perheelleni kannustuksesta ja siitä, että he ovat jaksaneet odottaa diplomityön valmistumista. Viimeisenä, muttei vähäisimpänä, haluan osoittaa kiitokseni yli 50 kaverille, jotka ovat antaneet henkistä tukea matkan varrella, sekä Jesaja 40:31:lle rohkaisevista sanoista.

Tampereella, 14. joulukuuta 2011

Hari Nortunen

SISÄLLYS

TERMIT JA SYMBOLIT

A	yleinen matriisi, tilamatriisi
A_c	laajennettu tilamatriisi $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$
B	yleinen matriisi, sisääntulomatriisi
C	lineaarisen yhtälöryhmän kerroinmatriisi, ulostulomatriisi
C^+	matriisin C transpoosi
C^*	matriisin C konjugaattitranspoosi
$\widetilde{C}^*$	differentiaaliyhtälöratkaisijan matriisiparametri
C^+	matriisin C pseudoinverssi
$C^{\sigma_1^2}$	skaalausmatriisi, joka saattaa tulon $\sigma_i\tilde{\sigma}_i$ arvoon σ_1^2
D	diagonaalimatriisi, myötäkytkentämatriisi
I_n	$n \times n$ -identiteettimatriisi
Im	imaginaariosa
L, U	LU-hajotelman matriisit, $C = LU$
N(μ, σ^2)	normaalijakauma, odotusarvona μ, varianssina σ^2
O	nollamatriisi
Re	reaaliosa
Tas(a, b)	jatkuva tasajakauma suljetulla välillä $[a, b]$
U, Λ, V	singulaariarvohajotelman matriisit, $C = U\Lambda V^*$
a_{ij}	matriisin A alkio rivin i sarakkeessa j
cond	matriisin kuntoluku
i, j	indeksi
i	imaginaariyksikkö
k	differentiaaliyhtälöratkaisijan parametri
$p \times n$	matriisin korkeus ja leveys
$\mathbf{r}$	lineaarisen yhtälöryhmän oikean puolen vektori
$\mathbf{r}_c$	laajennettu oikean puolen vektori $\begin{bmatrix} \mathbf{0} \\ -\mathbf{r} \end{bmatrix}$
randn(p, n)	satunnainen $p \times n$ -matriisi, jonka alkiot noudattavat N(0, 1) -normaalijakaumaa
rank	matriisin aste
s	sekunti
$\mathbf{u}(t)$	ajasta riippuva ohjausvektori
$\mathbf{x}$	vektori, lineaarisen yhtälöryhmän ratkaisuvektori
$\hat{\mathbf{x}}$	tasapainotila

$\mathbf{x}_c(t)$	laajennettu ajasta riippuva tilavektori $\begin{bmatrix}\mathbf{x}(t)\\ \xi(t)\end{bmatrix}$
$\mathbf{x}(t)$	ajasta riippuva tilavektori
α	differentiaaliyhtälöratkaisijan parametri
$\epsilon(t)$	differentiaaliyhtälön ratkaisun virhe ajan funktiona
ϵ_{tol}	toleranssi, jolla differentiaaliyhtälön ratkaisun virhe on riittävän pieni
λ, λ_i	ominaisarvo
$\xi(t)$	ajasta riippuva tilavektori
σ_i	singulaariarvo
$\tilde{\sigma}_i$	matriisin $\widetilde{C}^*$ singulaariarvo
$\lvert\cdot\rvert$	itseisarvot
$\lVert\cdot\rVert$	vektorin tai matriisin normi
$\langle\cdot, \cdot\rangle$	kahden vektorin sisätulo
$\blacksquare$	määritelmän loppu
$\square$	todistuksen loppu
$\lozenge$	esimerkin loppu

1. JOHDANTO

Lineaariset yhtälöt ovat tunnetusti hyvin tärkeitä. Jo peruskoulussa opitaan ratkaisemaan lineaarisia, ensimmäisen asteen yhtälöitä sekä yhden muuttujan tilanteessa että kahden muuttujan yhtälöparin tapauksessa. Ehkä yleisin tapa mallintaa reaalimaailman ilmiöitä on käyttää lineaarisia malleja. Vaikka selitettävä ilmiö olisi todellisuudessa epälineaarinen, mallia voidaan usein yksinkertaistaa sekä ymmärrettävyyden että laskennan kannalta approksimoimalla ilmiö lineaariseksi.

Oletetaan, että mallinnettavasta ilmiöstä on muodostettu lineaarinen yhtälöryhmä. Lineaarialgebrassa yhtälöryhmä esitetään tyypillisesti matriisimuodossa $C\mathbf{x} = \mathbf{r}$. Tarkastellaan tilannetta, jossa yhtälöryhmään kohdistuu pieniä muutoksia ajan kuluessa. Muutokset voivat kohdistua kerroinmatriisiin C tai vektoriin $\mathbf{r}$. Niiden muuttuessa myös ratkaisuvektori $\mathbf{x}$ muuttuu. Erityisesti jos C:n ja $\mathbf{r}$:n arvot saadaan mittaustuloksina, yhtälöryhmä muuttuu jokaisen mittauksen jälkeen, sillä mittaukset sisältävät aina virheitä. On selvää, että laskennallisesti joudutaan tekemään paljon ylimääräistä työtä, jos lineaarinen yhtälöryhmä on ratkaistava kokonaan uudelleen aina, kun kertoimet muuttuvat edes vähän. Mikäli muuttuneen yhtälöryhmän ratkaisu poikkeaa hyvin vähän muutosta edeltävän yhtälöryhmän ratkaisusta, olisi hyödyllistä, jos edellistä ratkaisua voisi jotenkin hyödyntää.

Tämän opinnäytetyön pääongelmana on muodostaa mahdollisimman nopea menetelmä, jolla voidaan ratkaista sarja lineaarisia yhtälöryhmiä $C_i\mathbf{x}_i = \mathbf{r}_i$, missä $i = 1, 2, \ldots, N$ ja N on suuri. Ratkaistavia yhtälöryhmiä on tässä opinnäytetyössä enintään tuhat ($N = 1000$), mutta sovelluksissa niitä voi olla tuhansia tai jopa miljoonia. Oletetaan, että peräkkäisissä yhtälöissä kerroinmatriisi C_i muuttuu vain vähän. Vaihtoehtoisesti myös oikean puolen vektoriin $\mathbf{r}_i$ voi kohdistua pieniä muutoksia. Tässä opinnäytetyössä pienillä muutoksilla tarkoitetaan sitä, että yksi tai useampi C_i:n (tai $\mathbf{r}_i$:n) alkio muuttuu siten, että alkioiden suhteelliset muutokset ovat pieniä. Esimerkiksi vektori $\begin{bmatrix}1 & 1\end{bmatrix}^T$ voi muuttua muotoon $\begin{bmatrix}1{,}001 & 0{,}998\end{bmatrix}^T$. Yhtälöryhmään kohdistuvia muutoksia kutsutaan **häiriöiksi**. Mainituista muutostyypeistä nimenomaan häiriöt kerroinmatriisissa vaikeuttavat yhtälöryhmien ratkaisua huomattavasti, koska matriisille C_i on haettava uudelleen käänteismatriisi jokaisen häiriön jälkeen — vaikka C_i muuttuisi vain vähän. Jos sen sijaan ainoastaan vektoria $\mathbf{r}_i$ häiritään ja matriisi C_i pysyy vakiona, yhtälöryhmien ratkaisu on lähes triviaalia: tällöin riittää kääntää C_i vain yhden kerran ja kertoa saatu inverssi N

kertaa erilaisilla vektoreilla $\mathbf{r}_i$. Koska vektorin $\mathbf{r}_i$ häiriöt on helppo käsitellä, opinnäytetyössä keskitytään pääasiassa tapaukseen, jossa kerroinmatriisiin C_i kohdistuu pieniä häiriöitä.

Yllä kuvattu ongelma saattaa esiintyä esimerkiksi sellaisessa prosessissa, jossa yhtälöryhmän kerroinmatriisi on käytännössä vakio, mutta todellisuudessa se muuttuu lievästi ajan funktiona. Eräs sovelluskohde voisi esimerkiksi olla, että yritys hankkii säännöllisin väliajoin 1000 eri tuotetta, joiden jokaisen kappalemäärä on tuntematon muuttuja. Jokaiseen tuotteeseen liittyy jokin tunnettu kappalehinta (yksikkönä euroa/kpl), ja ostoksiin on varattuna tietty rahamäärä sekä joitain muita erityisiä vaatimuksia. Ongelma on mallinnettu lineaarisen yhtälöryhmän avulla. Tuotteiden kappalehinnat muuttuvat muutamalla sentillä joka päivä, ja halutaan selvittää ostettavat kappalemäärät joka päivä 10 vuoden ajan. On selvää, ettei ole mielekästä kääntää kerroinmatriisia C_i noin 3650 kertaa. Sen sijaan olisi toivottavaa, että voitaisiin jotenkin hyödyntää edellisen päivän ratkaisua, koska muutaman sentin vaihtelut kappalehinnoissa eivät todennäköisesti muuta ostettavia kappalemääriä merkittävästi. Yleisen ongelman kannalta tavoitteena on päästä hyödyntämään edellisen yhtälöryhmän $C_{i-1}\mathbf{x}_{i-1} = \mathbf{r}_{i-1}$ ratkaisua siten, että häirityn yhtälöryhmän $C_i\mathbf{x}_i = \mathbf{r}_i$ ratkaisu voitaisiin helposti laskea edellisen ratkaisun avulla.

Pääongelman ratkaisemiseksi on esitetty tässä opinnäytetyössä differentiaaliyhtälöratkaisija, tai lyhennettynä **DY-ratkaisija**. Muodostetaan ensin lineaarinen differentiaaliyhtälösysteemi. Se voidaan esittää matriisimuodossa. Valitaan siinä esiintyvä matriisi siten, että systeemi on stabiili. Tämä tarkoittaa sitä, että millä tahansa alkuehdolla ratkaisu ohjautuu kohti jotain kiinteää pistettä, jota sanotaan tasapainotilaksi. Tämä tasapainotila voidaan selvittää. Muotoillaan lisäksi differentiaaliyhtälösysteemi siten, että lähestyessä tasapainotilaa ratkaisuvektorin yksi osa lähestyy lineaarisen yhtälöryhmän $C_i\mathbf{x}_i = \mathbf{r}_i$ ratkaisua. Simuloidaan differentiaaliyhtälöä esimerkiksi numeerisilla menetelmillä, kunnes osana systeemiä esiintyvä vektori $C_i\mathbf{x}_i(t) - \mathbf{r}_i$ on halutun toleranssin päässä nollavektorista. Sillä ajanhetkellä t_i laskettu ratkaisuvektori $\mathbf{x}_i(t_i)$ on lineaarisen yhtälöryhmän $C_i\mathbf{x}_i = \mathbf{r}_i$ numeerinen ratkaisu, ja se voidaan asettaa seuraavaa yhtälöryhmää $C_{i+1}\mathbf{x}_{i+1} = \mathbf{r}_{i+1}$ ratkaistaessa differentiaaliyhtälön alkutilaksi. Jos muutokset matriisien C_i ja C_{i+1} välillä oletetaan riittävän pieniksi, myös muutokset ratkaisujen $\mathbf{x}_i$ ja $\mathbf{x}_{i+1}$ välillä ovat melko pieniä. Tällöin uusi ratkaisu on tyypillisesti nopea laskea DY-ratkaisijalla edellistä ratkaisua alkutilana hyödyntäen. Todettakoon vielä, että DY-ratkaisijan ideana ei varsinaisesti ole ratkaista differentiaaliyhtälöä, vaan lineaarinen yhtälöryhmä $C_i\mathbf{x}_i = \mathbf{r}_i$. Menetelmän nimi "differentiaaliyhtälöratkaisija" tarkoittaa sitä, että (yhtälöryhmän) ratkaisu haetaan differentiaaliyhtälön avulla. Menetelmää ei tule sekoittaa differentiaaliyhtälö**n**ratkaisijoihin, esimerkiksi `ode45`, joskin kyseistä ratkaisijaa käytetään apuna simulaatioissa.

DY-ratkaisijan eräs tärkeä ominaisuus on **robustisuus**. Se tarkoittaa, että ratkaisija toimii, vaikka lineaarisen yhtälöryhmän kerroinmatriisia C_i tai vektoria $\mathbf{r}_i$ häiritään. Vaikka differentiaaliyhtälösysteemiin kohdistetaan muutoksia, ratkaisu voidaan edelleen hakea häiritylle systeemille. Differentiaaliyhtälöä ei tarvitse muotoilla uudelleen häiriöistä huolimatta. Tässä on merkittävä etu verrattuna perinteisiin lineaaristen yhtälöryhmien ratkaisijoihin, joiden on aina käännettävä C_i uudelleen kerroinmatriisin muuttuessa. DY-ratkaisija perustuu robustiin säätötekniikkaan. Yleisesti robustisuus on tärkeä ominaisuus säätötekniikassa. Vaikka tarkasteltavassa ilmiössä systeemiä häiritään, ratkaisun hakeminen on silti mahdollista, jos säätäjä on robusti. Joissain tapauksissa ilmiötä ei pystytä mallintamaan tarkasti tai mallia ei edes tiedetä tarkasti, mutta ratkaisu toimii edelleen, mikäli säätäjä on robusti.

Luvussa 2 käsitellään opinnäytetyön taustalla olevaa teoriaa pääpiirteittäin. Lineaarialgebran, differentiaaliyhtälöiden ja systeemitekniikan keskeisimmät käsitteet esitellään, joskin kaikkia tuloksia ei todisteta. Luvussa 3 määritellään jo johdannossa mainittu tutkimusongelma täsmällisesti, ja sen jälkeen muotoillaan tarkasteltava DY-ratkaisija teoreettiselta kannalta. Lopuksi suoritetaan hienosäätöä menetelmän parametreille tietokonelaskennan nopeuttamiseksi.

Luvussa 4 on ensin muotoiltu, millä periaatteella DY-ratkaisijaa verrataan muihin menetelmiin, ja seuraavaksi vertailu on toteutettu useiden simulaatioiden avulla. Vertauskohteeksi on valittu **LU-hajotelma**, joka on yleinen, tunnettu ja laskennallisesti pitkälle kehitetty ratkaisumenetelmä lineaarisille yhtälöryhmille. Kun yhtälöryhmän $C_i\mathbf{x}_i = \mathbf{r}_i$ kerroinmatriisiin kohdistetaan pääongelman oletuksen mukaisesti hyvin pieniä häiriöitä, DY-ratkaisija on tehokkaampi kuin LU-hajotelma. Luvun 4 viimeisessä esimerkissä lineaarinen yhtälöryhmä ratkaistaan 1000 kertaa C_i:n ollessa 4000×4000 -neliömatriisi. Huomataan, että DY-ratkaisija on lähes kaksi kertaa nopeampi kuin LU-hajotelma. Jos kerroinmatriisi C_i on leveä, eli matriisissa on enemmän sarakkeita kuin vaakarivejä, C_i:n kääntäminen vaikeutuu laskennallisesti, jolloin varsinkin suurilla matriiseilla LU-hajotelma hidastuu huomattavasti. Erikoisesti leveillä matriiseilla C_i differentiaaliyhtälösysteemi käyttäytyy vähemmän ongelmallisesti kuin neliömatriiseilla. Tällöin differentiaaliyhtälön simulointi ja sitä kautta yhtälöryhmän ratkaiseminen helpottuvat. Toisin sanottuna, leveät matriisit hidastavat LU-hajotelmaa mutta nopeuttavat DY-ratkaisijaa. Luvun 4 muutamassa esimerkissä on verrattu LU-hajotelmaa ja DY-ratkaisijaa kerroinmatriisin ollessa leveä. Havaitaan, että DY-ratkaisija on 5–10 kertaa nopeampi kuin LU-hajotelma.

DY-ratkaisija on vertauskohteeksi valittua LU-hajotelmaa tehokkaampi menetelmä opinnäytetyön pääongelman ratkaisemiseksi. Se ratkaisee nopeasti sarjan lineaarisia yhtälöryhmiä $C_i\mathbf{x}_i = \mathbf{r}_i$, kun muutokset peräkkäisten kerroinmatriisien (C_i ja C_{i+1}) tai oikean puolen vektorien ($\mathbf{r}_i$ ja $\mathbf{r}_{i+1}$) välillä ovat pieniä. Lisäksi DY-ratkaisijan etuna on robustisuus: riittää muodostaa kerran differentiaaliyhtälösys-

teemi vastaavalle lineaariselle yhtälöryhmälle $C_1\mathbf{x}_1 = \mathbf{r}_1$, minkä jälkeen ratkaisija toimii siitäkin huolimatta, että kerroinmatriisia C_i tai oikean puolen vektoria $\mathbf{r}_i$ häiritään.

2. TAUSTATIETOA

Esitellään ensin matriisilaskentaan liittyviä tuloksia, sitten differentiaaliyhtälöiden teoriaa ja lopuksi systeemiteoriaa. Tässä luvussa ja koko opinnäytetyössä viitataan monesti **The Mathworksin** kehittämään **MATLAB**-laskentaohjelmistoon [17], jolla muun muassa opinnäytetyön esimerkit ja simulaatiot on toteutettu.

2.1 Matriisilaskentaa

Matriisilaskennan ja lineaarialgebran käsittelyssä on seurattu lähinnä opintojakson **MAT-31090 Matriisilaskenta 1** opintomonistetta vuodelta 2008 (viite [22]). Jotkin tulokset on poimittu opintomonisteen vanhasta versiosta vuodelta 2005 (viite [21]), koska vuoden 2008 versiosta oli poistettu muutama opinnäytetyön kannalta tärkeä kohta, kuten lineaarisen yhtälöryhmän herkkyystarkastelu.

Tietokonelaskentaan kuluvaa aikaa arvioitiin ennen **flopsien** (*engl. FLOP, **Fl**oating-point **Op**eration*) avulla. Yksi flopsi koostuu yhdestä kertolaskusta sekä yhdestä yhteenlaskusta [22, s. 31]. Esimerkiksi skalaareilla a, b ja c operoidessa laskutoimitukseen $(a \cdot b + c)$ vaaditaan täsmälleen yksi flopsi. Laskennan työmäärää arvioidessa ollaan yleensä kiinnostuneita vain siitä, mitä suuruusluokkaa flopsien lukumäärä on. Jos flopsien lukumäärä on muotoa $\sum_{m=1}^{l} a_m n^m$, riittää useimmiten selvittää n:n korkein potenssi n^l suuruusluokan selvittämiseksi.

Esimerkki 2.1.1. [22, s. 31–32] Olkoon A ja B $n \times n$ -neliömatriiseja ja $\mathbf{x} \in \mathbb{R}^n$. Kertolaskuun

$$AB = \sum_{m=1}^{n} a_{im} b_{mj}, \qquad i = 1, \ldots, n, \quad j = 1, \ldots, n$$

tarvitaan $n \cdot n^2 = n^3$ flopsia. Tapauksessa $n = 1000$ laskemiseen tarvittavien flopsien lukumäärä on $n^3 = 10^9$, ja jos n on 10 000, flopseja tarvitaan vastaavasti 10^{12}. Jos arvioidaan, että yhden flopsin laskemiseen kuluu 10^{-6} sekuntia, matriisikertolaskut tapauksissa $n = 1000$ ja $n = 10\,000$ vievät aikaa 1000s ja 10 000s, vastaavasti. Matriisi-vektoritulossa $A\mathbf{x}$ tarvitaan $n \cdot n = n^2$ flopsia. $\Diamond$

Vanhoissa MATLAB-versioissa flopsien lukumäärän pystyi laskemaan komennolla `flops`. Kun numeerisen lineaarialgebran kirjasto **LAPACK** lisättiin MATLAB-

versioon 6, flopseja laskeva funktio otettiin pois käytöstä [17]. Uusissa versioissa laskennassa vaadittavaa työmäärää mitataan flopsien sijaan esimerkiksi tarkastelemalla **laskenta-aikoja** MATLAB-käskyillä `tic` ja `toc`. Tässä opinnäytetyössä flopseilla halutaan lähinnä havainnollistaa laskentatyön teoreettista määrää. Simulaatioissa sen sijaan tehty työ mitataan suoraan laskenta-aikoina.

Tarkastellaan seuraavaksi lineaarisia yhtälöryhmiä. Olkoon meillä n kappaletta muuttujia x_1, x_2, ..., x_n, jotka toteuttavat p kappaletta yhtälöitä:

$$\begin{cases} c_{11}x_1 + c_{12}x_2 + \ldots + c_{1n}x_n = r_1 \\ c_{21}x_1 + c_{22}x_2 + \ldots + c_{2n}x_n = r_2 \\ \qquad\qquad\qquad \vdots \\ c_{p1}x_1 + c_{p2}x_2 + \ldots + c_{pn}x_n = r_p \end{cases}, \tag{2.1}$$

missä $c_{ij} \in \mathbb{C}$ ja $r_i \in \mathbb{C}$, kun $i = 1, 2, \ldots, p$ ja $j = 1, 2, \ldots, n$. Teorian kannalta vakiot c_{ij} ja r_i oletetaan kompleksisiksi, jolloin ne ovat muotoa $a+ib$. Tämän opinnäytetyön esimerkeissä c_{ij} ja r_i ovat kuitenkin pääsääntöisesti reaalisia ($b = 0$), ellei toisin mainita. Yhtälö (2.1) voidaan kirjoittaa matriisi-vektoritulon avulla muodossa

$$C\mathbf{x} = \mathbf{r}, \tag{2.2}$$

missä

$$C = \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \ldots & c_{pn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{ja} \quad \mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix}.$$

Nyt C on kokoa $p \times n$ oleva **kerroinmatriisi**, $\mathbf{x}$ on kokoa $n \times 1$ oleva lineaarisen yhtälöryhmän **ratkaisuvektori** (tai lyhyemmin **ratkaisu**) ja $\mathbf{r}$ on kokoa $p \times 1$ oleva niin sanottu **oikean puolen vektori**. Tavallisesti C ja $\mathbf{r}$ on annettu, ja $\mathbf{x}$ halutaan selvittää. Tässä opinnäytetyössä tarkastellaan vain tapauksia $p \leq n$. Jos $p = n$, niin C on **neliömatriisi**, ja tapauksessa $p < n$ sanotaan, että C on **leveä matriisi**.

2.1.1 LU-hajotelma

Tietokonelaskennassa yksi tunnetuimmista ja käyttökelpoisimmista menetelmistä lineaarisen yhtälöryhmän ratkaisemiseksi on **LU-hajotelma**. Periaatteena on jakaa kerroinmatriisi C ala- ja yläkolmiomatriisin tuloksi siten, että

$$C = LU,$$

missä L on $p \times p$ -alakolmiomatriisi ja U $p \times n$ -yläkolmiomatriisi. Kun hajotelma on saatu muodostettua, sitä voidaan käyttää lineaarisen yhtälöryhmän ratkaisemisessa sekä kerroinmatriisin kääntämisessä. LU-hajotelma on käsitelty yksityiskohtaisesti viitteissä [7] ja [22, s. 52–66]. Kootaan keskeisimmät LU-hajotelmaan liittyvät tulokset jonkin verran yksinkertaistaen viitteestä [22].

Yksi tärkeimmistä LU-hajotelman käyttökohteista on lineaaristen yhtälöryhmien ratkaiseminen. Olkoon yhtälöryhmä muotoa $C\mathbf{x} = \mathbf{r}$, missä C on $n \times n$ -neliömatriisi. Muodostetaan ensin C:n LU-hajotelma $C = LU$, jolloin $C\mathbf{x} = LU\mathbf{x} = \mathbf{r}$. Merkitsemällä $U\mathbf{x} = \mathbf{y}$ lineaarinen yhtälöryhmä saadaan jaettua kahteen osaan:

$$\begin{cases} L\mathbf{y} = \mathbf{r} \\ U\mathbf{x} = \mathbf{y} \end{cases} . \tag{2.3}$$

Ensimmäiseltä riviltä ratkaistaan vektori $\mathbf{y}$, minkä jälkeen toiselta riviltä saadaan varsinainen ratkaisu $\mathbf{x}$. Koska L on muodostamistapansa perusteella ei-singulaarinen matriisi, yhtälöllä $L\mathbf{y} = \mathbf{r}$ on yksikäsitteinen ratkaisu $\mathbf{y} = L^{-1}\mathbf{r}$. Yhtälöllä $U\mathbf{x} = \mathbf{y}$ on yksikäsitteinen ratkaisu täsmälleen silloin, kun U on ei-singulaarinen. Matriisin U kääntyvyys taas on selvästi yhtäpitävä C:n kääntyvyyden kanssa.

Mikäli C on ei-singulaarinen neliömatriisi, sen käänteismatriisi voidaan laskea LU-hajotelman avulla. Merkitään toistaiseksi tuntematonta käänteismatriisia $C^{-1} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{bmatrix}$ ja identiteettimatriisia $I_n = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix}$. Ehdosta $CC^{-1} = I_n$ saadaan yhtälö $C \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix}$, ja tästä edelleen saadaan $\begin{bmatrix} C\mathbf{x}_1 & C\mathbf{x}_2 & \dots & C\mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_n \end{bmatrix}$. Nyt C:n käänteismatriisi voidaan muodostaa ratkaisemalla LU-hajotelman avulla n kappaletta yhtälöryhmiä $C\mathbf{x}_i = \mathbf{e}_i$, $i = 1, 2, \dots, n$.

Jos LU-hajotelma halutaan muodostaa ei-singulaariselle $n \times n$ -matriisille, laskentaan vaaditaan $\frac{1}{3}n^3$ flopsia. Kun $C = LU$, yhtälön $C\mathbf{x} = \mathbf{r}$ ratkaisu koostuu (2.3):n perusteella yhtälöiden $L\mathbf{y} = \mathbf{r}$ ja $U\mathbf{x} = \mathbf{y}$ ratkaisemisesta. Kumpaankin vaaditaan $\frac{1}{2}n^2$ flopsia, jolloin lineaarisen yhtälöryhmän ratkaisuun LU-hajotelman avulla kuluu yhteensä

$$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{2}n^2 = \frac{1}{3}n^3 + n^2 \tag{2.4}$$

flopsia. Mikäli yhtälöryhmä $C\mathbf{x} = \mathbf{r}$ halutaan ratkaista N:llä eri vektorilla $\mathbf{r}$, LU-hajotelma riittää laskea vain kerran, mutta yhtälöt $L\mathbf{y} = \mathbf{r}$ ja $U\mathbf{x} = \mathbf{y}$ on ratkaistava uudelleen yhteensä N kertaa. Tällöin laskentaan tarvitaan $\frac{1}{3}n^3 + N \cdot n^2$ flopsia. Tämä on myös se flopsien määrä, joka vaaditaan, jos yhtälöryhmässä vektoria $\mathbf{r}$ häiritään N kertaa ja halutaan ratkaista N kappaletta häirittyjä yhtälöitä $C\mathbf{x} = \tilde{\mathbf{r}}$. Käänteismatriisin C^{-1} muodostamisessa tarvitaan n kappaleita vektoreita $\mathbf{e}_i$, $i = 1, 2, \dots, n$. Tällöin laskennassa vaadittavien flopsien lukumäärä $\frac{1}{3}n^3 + n \cdot n^2 = \frac{4}{3}n^3$. Jos yhtälöryhmä $C\mathbf{x} = \mathbf{r}$ halutaan ratkaista N:llä eri matriisilla C, LU-hajotelma

on muodostettava jokaisella eri matriisilla kokonaan uudelleen, ja samoin yhtälöt $L\mathbf{y} = \mathbf{r}$ ja $U\mathbf{x} = \mathbf{y}$ on ratkaistava uudelleen. LU-hajotelma on siis erittäin herkkä C:n häiriöille. Mikäli kerroinmatriisia häiritään N kertaa ja halutaan ratkaista N kappaletta häirittyjä yhtälöitä $\widetilde{C}\mathbf{x} = \mathbf{r}$, yhtälöiden ratkaisemiseen vaaditaan $N \cdot \left(\frac{1}{3}n^3 + n^2\right)$ flopsia.

MATLAB käyttää LU-hajotelman muodostamiseen LAPACK-rutiineja [1]. Mikäli C on harva matriisi ja komennolle `lu` annetaan vähintään 4 ulostuloparametria, MATLAB käyttää UMFPACK-rutiineja [4]. Yhtälö $C\mathbf{x} = \mathbf{r}$ voidaan ratkaista MATLAB-käskyllä `x = mldivide(C,r)`, tai lyhyemmin `x = C\r`. Yhtälön ratkaisemisessa käytetään muun muassa LU-hajotelmaa. Jos C on neliömatriisi, käsky `C\r` tekee käytännössä saman kuin komento `inv(C)*r` [1] (eli $C^{-1}\mathbf{r}$), paitsi että edellinen hakee ratkaisun **Gaussin eliminaatiolla** kääntämättä suoraan matriisia C. Jos C on leveä $p \times n$ -matriisi, $p < n$, ja C:llä on täysi aste p, käsky `x = C\r` hakee ratkaisun $\mathbf{x}$ siten, että sillä on enintään p kappaletta nollasta eroavia alkioita. Huomaa, että saatu ratkaisu ei yleensä ole sama kuin käskyllä `pinv(C)*r` (eli $C^{+}\mathbf{r}$, missä C^{+} on C:n pseudoinverssi) saatava **pienimmän neliösumman ratkaisu** [1, 16], jossa minimoidaan normia $\|C\mathbf{x} - \mathbf{r}\|$ [17]. Lisäksi singulaariarvohajotelman yhteydessä selviää, että käskyllä `pinv(C)*r` normi $\|\mathbf{x}\|$ minimoituu. Singulaariarvohajotelmaan ja pseudoinverssiin palataan kappaleessa 2.1.3.

Esimerkki 2.1.2. Tarkastellaan lineaarista yhtälöryhmää $C\mathbf{x} = \mathbf{r}$, missä

$$C = \begin{bmatrix} 1 & 1 & 1 & 2 \\ -1 & 2 & 1 & 2 \\ 1 & 4 & 2 & 1 \end{bmatrix} \quad \text{ja} \quad \mathbf{r} = \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix}.$$

Matriisille C voidaan muodostaa LU-hajotelma seuraavalla tavalla:

$$C = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & 3 & 2 & 4 \\ 0 & 0 & -1 & -5 \end{bmatrix} = LU.$$

Merkitään nyt $U\mathbf{x} = \mathbf{y}$, jolloin ratkaistavana on (2.3):n perusteella yhtälöt $L\mathbf{y} = \mathbf{r}$ ja $U\mathbf{x} = \mathbf{y}$. Kun yhtälö $L\mathbf{y} = \mathbf{r}$ kirjoitetaan alkioittain,

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix},$$

ratkaisu on helppo nähdä, kun aloitetaan ensimmäisestä rivistä ja edetään “ylhäältä

alaspäin":

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 3 \end{bmatrix}^T.$$

Lopuksi on ratkaistavana yhtälö $U\mathbf{x} = \mathbf{y}$, joka on alkioittain kirjoitettuna

$$\begin{bmatrix} 1 & 1 & 1 & 2 \\ 0 & 3 & 2 & 4 \\ 0 & 0 & -1 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}.$$

Koska leveän matriisin tapauksessa mukana on enemmän muuttujia kuin yhtälöitä, yhtälöllä $U\mathbf{x} = \mathbf{y}$ ja siten myös alkuperäisellä yhtälöryhmällä $C\mathbf{x} = \mathbf{r}$ on ääretön määrä ratkaisuja. Valitaan x_4 niin sanotuksi vapaaksi muuttujaksi: $x_4 = t \in \mathbb{R}$. Aloittamalla viimeisestä rivistä ja etenemällä "alhaalta ylöspäin" ratkaisuksi saadaan suora

$$\mathbf{x} = \begin{bmatrix} 2 \\ 2 \\ -3 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 2 \\ -5 \\ 1 \end{bmatrix}, \quad t \in \mathbb{R}. \tag{2.5}$$

MATLAB-käsky `x = C\r` antaa yksittäisen ratkaisun siten, että mahdollisimman moni $\mathbf{x}$:n alkioista on nollia. Koska $\operatorname{rank}(C) = p = 3$, enintään 3 kappaletta $\mathbf{x}$:n alkioista on nollasta eroavia. MATLAB:in antama ratkaisu on

$$\mathbf{x} = \begin{bmatrix} 7/5 \\ 4/5 \\ 0 \\ -3/5 \end{bmatrix}, \tag{2.6}$$

mikä saadaan täydellisestä ratkaisusta (2.5) valitsemalla $t = -\frac{3}{5}$. Jatkossa tyydytään hakemaan yhtälöryhmälle $C\mathbf{x} = \mathbf{r}$ jokin yksittäinen ratkaisu. ◊

2.1.2 Ominaisarvot ja -vektorit

Tässä kappaleessa tarkastellaan neliömatriisin A ominaisarvoja ja -vektoreita. Samalla esitellään muun muassa definiittisyyden käsite.

Määritelmä 2.1.3. Olkoon A $n \times n$ -matriisi. Etsitään skalaaria $\lambda \in \mathbb{C}$ ja vektoria $\mathbf{x} \in \mathbb{C}^n$ siten, että ne toteuttavat yhtälön

$$A\mathbf{x} = \lambda\mathbf{x}, \quad \mathbf{x} \neq \mathbf{0}. \tag{2.7}$$

Yhtälön (2.7) toteuttavaa skalaaria λ sanotaan A:n **ominaisarvoksi**, ja λ:a vastaavaa ratkaisuvektoria $\mathbf{x}$ sanotaan ominaisarvoa λ vastaavaksi **ominaisvektoriksi**. ■

Ominaisarvojen ratkaisemiseksi yhtälö (2.7) voidaan saattaa ekvivalenttiin muotoon:

$$\begin{aligned} & \exists \mathbf{x} \neq \mathbf{0}: && A\mathbf{x} - \lambda\mathbf{x} = \mathbf{0} && \\ \iff\ & \exists \mathbf{x} \neq \mathbf{0}: && (A - \lambda I)\mathbf{x} = \mathbf{0} && \text{(2.8a)} \\ \iff\ & && \det(A - \lambda I) = 0. && \text{(2.8b)} \end{aligned}$$

Saadaan siis determinanttiyhtälö, jossa ominaisvektorit eivät ole mukana. Kun ominaisarvot on selvitetty yhtälöstä (2.8b), ne voidaan sijoittaa yhtälöön (2.8a), josta taas ominaisvektorit voidaan ratkaista.

Määritelmä 2.1.4. Kokoa $n \times n$ olevat matriisit A ja B ovat **similaarisia**, jos on olemassa ei-singulaarinen $n \times n$ -matriisi S siten, että $B = S^{-1}AS$. Jos A on similaarinen diagonaalimatriisin kanssa, niin A on **diagonalisoituva**. ■

Similaarisilla matriiseilla on sellainen tärkeä ominaisuus, että niillä on samat ominaisarvot.

Määritelmä 2.1.5. Kokoa $n \times n$ oleva matriisi A on **unitaarinen**, jos $AA^* = A^*A = I_n$, missä A^* on matriisin A **konjugaattitranspoosi**. ■

Toisin sanottuna, A on unitaarinen jos ja vain jos $A^{-1} = A^*$.

Määritelmä 2.1.6. Kokoa $n \times n$ olevat matriisit A ja B ovat **unitaarisesti similaarisia**, jos on olemassa unitaarinen matriisi U siten, että $B = U^*AU$. ■

Kootaan tähän muutama diagonalisoituvuuteen liittyvä tulos. Niiden todistukset sivuutetaan.

Lause 2.1.7. *Matriisi A on unitaarisesti similaarinen diagonaalimatriisin D kanssa jos ja vain jos A on **normaali**: $A = UDU^* \iff AA^* = A^*A$, missä U on unitaarinen matriisi.*

Lause 2.1.8. *Kokoa $n \times n$ oleva matriisi A on similaarinen diagonaalimatriisin D kanssa jos ja vain jos sillä on n kappaletta lineaarisesti riippumattomia ominaisvektoreita. Tällöin*

$$A = QDQ^{-1},$$

missä ei-singulaarinen matriisi Q koostuu A:n lineaarisesti riippumattomista ominaisvektoreista $\mathbf{v}_i$, $i = 1, 2, \ldots, n$ siten, että $Q = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$. Matriisin D diagonaalialkiot ovat A:n vastaavat ominaisarvot: $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Lause 2.1.9. *Olkoon A kokoa $n \times n$ ja* ***hermiittinen*** *(*engl. Hermitian*): $A^* = A$. Tällöin A:lla on täsmälleen n kappaletta lineaarisesti riippumattomia ominaisvektoreita $\mathbf{x}_i$, $i = 1, 2, \ldots, n$. Ominaisvektorit voidaan valita siten, että niistä voi koota unitaarisen matriisin $U = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$, ja*

$$A = UDU^*,$$

missä D koostuu A:n ominaisarvoista: $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Ennen definiittisyyden määrittelyä esitellään sisätulon käsite:

Määritelmä 2.1.10. Vektoreiden $\mathbf{x} \in \mathbb{C}^n$ ja $\mathbf{y} \in \mathbb{C}^n$ välinen **sisätulo** määritellään kaavalla

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} \overline{x}_i y_i, \tag{2.9}$$

missä $\overline{x}_i$ on x_i:n liittoluku. ■

Mainitsemisen arvoinen on myös sisätulon ja matriisin konjugaattitranspoosin välillä oleva yhteys:

$$\langle \mathbf{x}, A\mathbf{y} \rangle = \langle A^*\mathbf{x}, \mathbf{y} \rangle \quad \forall \mathbf{x}, \mathbf{y}. \tag{2.10}$$

Määritelmä 2.1.11. Hermiittinen $n \times n$ -matriisi A on

1. **positiivisesti definiitti**, jos $\langle \mathbf{x}, A\mathbf{x} \rangle > 0 \ \forall \mathbf{x} \neq \mathbf{0}$.
2. **positiivisesti semidefiniitti**, jos $\langle \mathbf{x}, A\mathbf{x} \rangle \geq 0 \ \forall \mathbf{x}$.
3. **negatiivisesti definiitti**, jos $\langle \mathbf{x}, A\mathbf{x} \rangle < 0 \ \forall \mathbf{x} \neq \mathbf{0}$.
4. **negatiivisesti semidefiniitti**, jos $\langle \mathbf{x}, A\mathbf{x} \rangle \leq 0 \ \forall \mathbf{x}$.
5. **indefiniitti**, jos A ei ole positiivisesti eikä negatiivisesti semidefiniitti.

■

Hermiittisen matriisin definiittisyys on helppo selvittää ominaisarvojen avulla [13, 22]:

Lause 2.1.12. *Hermiittinen $n \times n$ -matriisi A on*

1. *positiivisesti definiitti $\iff$ A:n ominaisarvot ovat positiivisia.*
2. *positiivisesti semidefiniitti $\iff$ A:n ominaisarvot ovat ei-negatiivisia.*
3. *negatiivisesti definiitti $\iff$ A:n ominaisarvot ovat negatiivisia.*
4. *negatiivisesti semidefiniitti $\iff$ A:n ominaisarvot ovat ei-positiivisia.*
5. *indefiniitti $\iff$ A:lla on sekä positiivisia että negatiivisia ominaisarvoja.*

Todistus. Katso [22, s. 127–128]. □

2.1.3 Singulaariarvohajotelma

Olkoon C matriisi, jonka aste on täysi. Pyritään saattamaan C diagonaaliseen muotoon unitaarisilla muunnosmatriiseilla. Seuraavassa lauseessa muodostetetaan C:lle niin sanottu **singulaariarvohajotelma** (*engl. singular value decomposition, lyh. SVD*) [22, s. 146–153].

Lause 2.1.13. *Olkoon C $p \times n$ -matriisi siten, että* $\operatorname{rank}(C) = p \leq n$. *On olemassa unitaariset kokoa $p \times p$ ja $n \times n$ olevat matriisit U ja V siten, että*

$$C = U\Lambda V^*, \tag{2.11}$$

missä Λ on $p \times n$ -diagonaalimatriisi siten, että

$$\Lambda = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p) = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & \sigma_p & \ldots & 0 \end{bmatrix},$$

ja $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$. Lauseketta (2.11) sanotaan C:n singulaariarvohajotelmaksi, ja Λ:n diagonaalialkioita σ_i sanotaan C:n singulaariarvoiksi, $i = 1, 2, \ldots, p$.

Todistus. Katso [22, s. 146–147]. □

MATLAB käyttää singulaariarvohajotelman muodostamisessa LAPACK-rutiineja [1]. Käsky `s = svd(C)` palauttaa C:n singulaariarvot vektorina, ja hieman pidempi käsky `[U,S,V] = svd(C)` antaa koko singulaariarvohajotelman $C = U\Lambda V^*$. Koko hajotelman muodostamiseen vaadittava laskentatyö on $n \times n$ -matriisin tapauksessa likimäärin $13n^3$ flopsia [22, s. 149].

Määritelmä 2.1.14. Kokoa $p \times n$ olevan matriisin C **matriisinormi** määritellään seuraavilla lausekkeilla, jotka ovat keskenään yhtäpitävät [22, s. 41–42]:

$$\|C\| = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|C\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|\leq 1} \|C\mathbf{x}\| = \max_{\|\mathbf{x}\|=1} \|C\mathbf{x}\| . \tag{2.12}$$

■

Lause 2.1.15. *Matriisinormilla on voimassa seuraavat ominaisuudet:*

1. $\|A\| \geq 0; \quad \|A\| = 0 \iff A = O$
2. $\|\alpha A\| = |\alpha| \, \|A\| \quad \forall \alpha \in \mathbb{C}$
3. $\|AB\| \leq \|A\| \, \|B\|$
4. $\|A + B\| \leq \|A\| + \|B\|$,

missä A ja B ovat $p \times n$ -matriiseja.

Todistus. Katso [22, s. 42]. □

Matriisinormin laskeminen lausekkeiden (2.12) avulla on yleensä hankalaa. Singulaariarvohajotelman avulla matriisinormin laskeminen helpottuu huomattavasti.

Lause 2.1.16. *Olkoon C kokoa $p \times n$ ja sen singulaariarvot σ_1, σ_2, …, σ_p. Matriisinormille on voimassa*

$$\|C\| = \max_{i=1,\ldots,p} \sigma_i = \sigma_1. \tag{2.13}$$

Jos C on lisäksi ei-singulaarinen neliömatriisi ($p = n$), niin

$$\left\|C^{-1}\right\| = \frac{1}{\min\limits_{i=1,\ldots,p} \sigma_i} = \frac{1}{\sigma_n}. \tag{2.14}$$

Todistus. Katso [22, s. 151–152]. □

Tarkastellaan seuraavaksi lineaarisen yhtälöryhmän $C\mathbf{x} = \mathbf{r}$ ratkaisemista singulaariarvohajotelman avulla, kun C on kokoa $p \times n$ [22, s. 156–161]. Esitellään ensin **pseudoinverssin** käsite.

Määritelmä 2.1.17. Olkoon $p \times n$ -matriisin C singulaariarvohajotelma muotoa

$$C = U\Lambda V^*,$$

missä

$$\Lambda = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_p & \dots & 0 \end{bmatrix}.$$

Matriisin C pseudoinverssi on C^+ on $n \times p$ -matriisi siten, että

$$C^+ = V\Lambda^+ U^*. \tag{2.15}$$

missä

$$\Lambda^+ = \begin{bmatrix} 1/\sigma_1 & 0 & \dots & 0 \\ 0 & 1/\sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/\sigma_p \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

Matriisi Λ^+ muodostetaan siis hakemalla Λ:n transpoosi ja korvaamalla nollasta eroavat alkiot niiden käänteisluvuillaan. ■

Jos C on leveä matriisi ($p < n$), jonka aste on täysi p, niin C:llä on **oikeanpuoleinen inverssi** ja $CC^+ = I_p$. Jos taas C on korkea matriisi ($p > n$), jonka aste on täysi n, niin C:llä on **vasemmanpuoleinen inverssi** ja $C^+C = I_n$. Mikäli C on neliömatriisi, jonka aste on täysi, sillä on sekä vasen että oikea inverssi ja $C^+ = C^{-1}$. Tällöin C:n käänteismatriisi voidaan muodostaa singulaariarvohajotelman avulla seuraavalla tavalla: $C^{-1} = (U\Lambda V^*)^{-1} = V\Lambda^{-1}U^*$.

Ennen seuraavaa lausetta määritellään yleisen matriisin C **nolla-avaruus** eli **ydin**:

$$\mathcal{N}(C) = \{\mathbf{x} \mid C\mathbf{x} = \mathbf{0}\}.$$

Lause 2.1.18. *Olkoon C kokoa $p \times n$ ja* $\operatorname{rank}(C) = p$.

1. *Jos $p < n$, niin yhtälöryhmällä $C\mathbf{x} = \mathbf{r}$ on äärettömän monta ratkaisua, jotka ovat muotoa*
$$\mathbf{x} = C^{+}\mathbf{r} + \mathbf{v}, \quad \textit{missä} \quad \mathbf{v} \in \mathcal{N}(C).$$

2. *Jos $p = n$, niin yhtälöryhmällä on yksikäsitteinen ratkaisu $\mathbf{x} = C^{-1}\mathbf{r}$.*

Todistus. Todistetaan ensin kohta 1, minkä jälkeen kohdan 2 todistaminen on helppoa.

1. Matriisilla C on oikeanpuoleinen inverssi, joten $CC^{+} = I_p$. Tällöin
$$\begin{aligned} C\mathbf{x} = \mathbf{r} &\iff C\mathbf{x} = CC^{+}\mathbf{r} \\ &\iff C(\mathbf{x} - C^{+}\mathbf{r}) = \mathbf{0} \\ &\iff \mathbf{x} - C^{+}\mathbf{r} \in \mathcal{N}(C) \\ &\iff \mathbf{x} - C^{+}\mathbf{r} = \mathbf{v} \in \mathcal{N}(C) \\ &\iff \mathbf{x} = C^{+}\mathbf{r} + \mathbf{v}, \quad \mathbf{v} \in \mathcal{N}(C). \end{aligned}$$

2. Koska $\operatorname{rank}(C) = p = n$, niin $\mathcal{N}(C) = \{\mathbf{0}\}$, jolloin $\mathbf{v} = \mathbf{0}$. Tällöin yhtälöryhmällä on yksikäsitteinen ratkaisu $\mathbf{x} = C^{+}\mathbf{r}$. Koska C^{+} on sekä vasen että oikea inverssi, niin $C^{+} = C^{-1}$.

□

Kohdassa 1 saatu ratkaisu ei ole yksikäsitteinen. Mikäli halutaan minimoida normi $\|\mathbf{x}\| = \|C^{+}\mathbf{r} + \mathbf{v}\|$, tämä onnistuu valitsemalla $\mathbf{v} = \mathbf{0}$ (todistus viitteessä [22, s. 160–161]), jolloin **miniminormiratkaisuna** on $\mathbf{x} = C^{+}\mathbf{r}$.

2.1.4 Lineaarisen yhtälöryhmän ratkaisun herkkyydestä

Tarkastellaan, miten lineaarisen yhtälöryhmän ratkaisu muuttuu, kun kerroinmatriisia C häiritään. Rajoitutaan tarkastelemaan $n \times n$ -neliömatriiseja [21, s. 172–177].

Määritelmä 2.1.19. Olkoon C $n \times n$ -neliömatriisi. Sen **kuntoluku** (*engl. condition number*) on
$$\operatorname{cond}(C) = \frac{\sigma_1}{\sigma_n}, \tag{2.16}$$
missä σ_1 ja σ_n ovat C:n suurin ja pienin singulaariarvo. ■

Ei-singulaarisella matriisilla $\sigma_n > 0$, ja koska $\sigma_1 \geq \sigma_n$, niin kääntyvälle matriisille on voimassa $1 \leq \operatorname{cond}(C) < \infty$. Kuntoluku kuvaa, kuinka vaikeaa C:n kääntäminen on numeeriselta kannalta. Mikäli $\operatorname{cond}(C)$ on suuri, C:n inverssin laskeminen on numeerisesti vaikeaa. Jos taas $\operatorname{cond}(C) \approx 1$, kääntäminen on numeerisesti helppoa.

Voidaan osoittaa, että unitaarisella matriisilla $\sigma_1 = \sigma_2 = \ldots = \sigma_n = 1$, jolloin $\text{cond}(C) = \frac{\sigma_1}{\sigma_n} = 1$. Unitaarisen matriisin kääntäminen on siis helppoa numeeriselta kannalta.

Jos C on ei-singulaarinen $n \times n$ -neliömatriisi, yhtälöryhmällä $C\mathbf{x} = \mathbf{r}$ on yksikäsitteinen ratkaisu $\mathbf{x} = C^{-1}\mathbf{r}$. Oletetaan, että C:tä häiritään siten, että $C \to C + \Delta C$. Merkitään häirittyä ratkaisua $\mathbf{x} + \Delta\mathbf{x}$. Jos häiritty matriisi $C + \Delta C$ on ei-singulaarinen, niin

$$\mathbf{x} + \Delta\mathbf{x} = (C + \Delta C)^{-1}\mathbf{r},$$

ja tästä edelleen saadaan [21, s. 175]:

$$\Delta\mathbf{x} = (C + \Delta C)^{-1}\mathbf{r} - C^{-1}\mathbf{r} = \left[(C + \Delta C)^{-1} - C^{-1}\right]\mathbf{r}.$$

Koska viitteen [21, s. 175] nojalla on voimassa

$$(C + \Delta C)^{-1} - C^{-1} = -C^{-1}(\Delta C)(C + \Delta C)^{-1},$$

niin edelleen

$$\Delta\mathbf{x} = -C^{-1}(\Delta C)(C + \Delta C)^{-1}\mathbf{r} = -C^{-1}(\Delta C)(\mathbf{x} + \Delta\mathbf{x}),$$

ja ratkaisun muutoksen suuruudelle saadaan arvio, kun hyödynnetään matriisinormin ominaisuuksia (Lause 2.1.15) sekä Lauseen 2.1.16 tuloksia:

$$\begin{aligned}
\|\Delta\mathbf{x}\| &\leq \left\|C^{-1}\right\| \|\Delta C\| \|\mathbf{x} + \Delta\mathbf{x}\| \\
&= \|C\| \left\|C^{-1}\right\| \frac{\|\Delta C\|}{\|C\|} \|\mathbf{x} + \Delta\mathbf{x}\| \\
&= \sigma_1 \frac{1}{\sigma_n} \frac{\|\Delta C\|}{\|C\|} \|\mathbf{x} + \Delta\mathbf{x}\| \\
&= \text{cond}(C) \frac{\|\Delta C\|}{\|C\|} \|\mathbf{x} + \Delta\mathbf{x}\| .
\end{aligned}$$

Mikäli $\Delta\mathbf{x}$ oletetaan pieneksi $\mathbf{x}$:ään verrattuna, likimäärin on voimassa:

$$\|\Delta\mathbf{x}\| \leq \text{cond}(C) \frac{\|\Delta C\|}{\|C\|} \|\mathbf{x}\| . \tag{2.17}$$

Olkoon C edelleen kokoa $n \times n$ ja ei-singulaarinen. Yhtälöryhmän $C\mathbf{x} = \mathbf{r}$ yksikäsitteinen ratkaisu on $\mathbf{x} = C^{-1}\mathbf{r}$. Oletetaan nyt, että vektoria $\mathbf{r}$ häiritään siten, että $\mathbf{r} \to \mathbf{r} + \Delta\mathbf{r}$. Merkitään häirittyä ratkaisua jälleen $\mathbf{x} + \Delta\mathbf{x}$, jolloin

$$C(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{r} + \Delta\mathbf{r},$$

ja tästä saadaan edelleen [22, s. 173–174]:

$$\mathbf{x} + \Delta\mathbf{x} = C^{-1}\mathbf{r} + C^{-1}\Delta\mathbf{r}.$$

Koska $\mathbf{x} = C^{-1}\mathbf{r}$, on oltava

$$\Delta\mathbf{x} = C^{-1}\Delta\mathbf{r},$$

ja tästä saadaan matriisinormin ominaisuuksien avulla arvio:

$$\|\Delta\mathbf{x}\| = \left\|C^{-1}\Delta\mathbf{r}\right\| \leq \left\|C^{-1}\right\| \|\Delta\mathbf{r}\| = \frac{1}{\sigma_p} \|\Delta\mathbf{r}\|.$$

Kertomalla ja jakamalla vektorin $\mathbf{r}$ pituudella saadaan:

$$\|\Delta\mathbf{x}\| \leq \frac{1}{\sigma_p} \frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \|\mathbf{r}\|,$$

ja koska $\mathbf{r} = C\mathbf{x}$ ja edelleen $\|\mathbf{r}\| \leq \|C\| \|\mathbf{x}\| = \sigma_1 \|\mathbf{x}\|$, niin tästä saadaan:

$$\begin{aligned} \|\Delta\mathbf{x}\| &\leq \frac{1}{\sigma_p} \frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \|\mathbf{r}\| \\ &\leq \frac{1}{\sigma_p} \frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \sigma_1 \|\mathbf{x}\| \\ &= \frac{\sigma_1}{\sigma_p} \frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \|\mathbf{x}\|. \end{aligned}$$

Siis ratkaisun muutoksen $\Delta\mathbf{x}$ suuruudelle saadaan yläraja:

$$\|\Delta\mathbf{x}\| \leq \operatorname{cond}(C) \frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \|\mathbf{x}\|. \tag{2.18}$$

Esimerkki 2.1.20. Olkoon C kokoa $n \times n$ oleva **Hilbertin matriisi** [8]:

$$c_{ij} = \frac{1}{i+j-1}, \quad \text{tai alkioittain} \quad C = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}.$$

Tapauksessa $n = 10$ Hilbertin matriisin kuntoluku on $\operatorname{cond}(C) = 1{,}6025 \cdot 10^{13}$. Valitaan $\mathbf{x} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$, jolloin vektori $\mathbf{r}$ saadaan laskettua [21, s. 177]: $\mathbf{r} = C\mathbf{x}$. Kohdistetaan seuraavaksi vektoriin $\mathbf{r}$ pieni häiriö:

$$\mathbf{r} \to \mathbf{r} + 0{,}001 \cdot \operatorname{randn}(10,\, 1),$$

missä merkintä randn(10, 1) := $\Delta\mathbf{r}$ tarkoittaa satunnaista, N(0, 1)-normaalijakautunutta 10×1 -vektoria. Nyt $\frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \propto 10^{-3}$, mutta kun lasketaan uusi ratkaisu

$$\tilde{\mathbf{x}} =: \mathbf{x} + \Delta\mathbf{x} = C^{-1}(\mathbf{r} + \Delta\mathbf{r}),$$

huomataan, että ratkaisun suhteellinen muutos on $\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \propto 10^{9}$. Hilbertin matriisi on siis erittäin herkkä häiriöille. Vastaavasti teorian (kaava (2.18)) perusteella saadaan

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathrm{cond}(C)\frac{\|\Delta\mathbf{r}\|}{\|\mathbf{r}\|} \propto 10^{13} \cdot 10^{-3} = 10^{10}.$$

Simulaatioissa päädyttiin siis lähes huonoimpaan mahdolliseen tilanteeseen. ◊

2.2 Differentiaaliyhtälöt

Differentiaaliyhtälössä on kyse yhtälöstä, jossa esiintyy tuntemattomia funktioita sekä niiden derivaattoja. Mikäli näissä derivaatoissa esiintyy vain "tavallisia" derivaattoja (ei osittaisderivaattoja), kyseessä on **(tavallinen) differentiaaliyhtälö**, tai lyhennettynä **DY** (*engl. ordinary differential equation, ODE*). Tässä opinnäytetyössä tarkastellaan ainoastaan tavallisia differentiaaliyhtälöitä. Niiden käsittelyssä seurataan lähinnä viitteitä [9] ja [28].

Differentiaaliyhtälön **kertaluku** määräytyy siinä esiintyvän korkeimman derivaatan kertaluvun perusteella. Jos tuntematon funktio ja sen derivaatat esiintyvät yhtälössä lineaarisesti (ei esimerkiksi potenssiin $n \neq 1$ korotettuna tai trigonometrisessä funktiossa), differentiaaliyhtälöä sanotaan **lineaariseksi**. Mikäli selvitettävän funktion ja sen derivaattojen kertoimet ovat vakioita, DY on **vakiokertoiminen**. Tässä opinnäytetyössä tarkastellaan 1. kertaluvun lineaarisia ja vakiokertoimisia differentiaaliyhtälöitä. Niitä ratkaistaessa tuntematon funktio ja sen derivaatat siirretään yleensä yhtälön vasemmalle puolelle. Jos tämän jälkeen oikealle puolelle ei jää mitään termejä, kyseessä on **homogeeninen yhtälö**. Muussa tapauksessa kyseessä on **epähomogeeninen yhtälö**.

Tyypillisin differentiaaliyhtälö on muotoa

$$x'(t) = ax(t), \tag{2.19}$$

missä $x(t)$ on tuntematon funktio, joka riippuu muuttujasta t, $x'(t)$ on funktion ensimmäinen derivaatta ja $a \in \mathbb{R}$ on vakio. Jatkossa muuttuja t tarkoittaa **aikaa**, jos ei toisin mainita. Helposti nähdään, että yhtälön (2.19) **yleinen ratkaisu** on $x(t) = \mathrm{e}^{at}c$, missä $c \in \mathbb{R}$ on mielivaltainen vakio. Jos c halutaan kiinnittää, tämä onnistuu **alkuehdoilla**, jotka ovat tyypillisesti muotoa $x(0) = x_0$, missä $x_0 \in \mathbb{R}$ tunnetaan. Differentiaaliyhtälöt esitetään usein **alkuarvoprobleemana** (*engl. initial*

value problem):

$$x'(t) = ax(t), \quad x(0) = x_0. \tag{2.20}$$

Nyt ratkaisun $x(t)$ täytyy toteuttaa yhtälö (2.19), ja sen lisäksi funktion on saatava arvo x_0 ajanhetkellä $t = 0$. Yleisestä ratkaisusta $x(t) = \mathrm{e}^{at}c$ ja alkuehdosta $x(0) = x_0$ saadaan, että $x(0) = \mathrm{e}^{a\cdot 0}c = c = x_0$, jolloin differentiaaliyhtälön ratkaisu on $x(t) = \mathrm{e}^{at}x_0$. Voidaan osoittaa, että saatu ratkaisu on yksikäsitteinen.

Mikäli vakiokerroin $a \in \mathbb{R}$, yhtälössä (2.19) a:n etumerkki määrää, miten ratkaisu $x(t) = \mathrm{e}^{at}c$ käyttäytyy ajan lähestyessä ääretöntä:

1. jos $a > 0$, niin $\lim_{t\to\infty} \mathrm{e}^{at}c = \infty$, kun $c > 0$, ja $\lim_{t\to\infty} \mathrm{e}^{at}c = -\infty$, kun $c < 0$.
2. jos $a = 0$, niin $\mathrm{e}^{at}c$ on vakio.
3. jos $a < 0$, niin $\lim_{t\to\infty} \mathrm{e}^{at}c = 0$.

Tarkastellaan seuraavaksi ensimmäisen kertaluvun lineaarista, epähomogeenista ja vakiokertoimista differentiaaliyhtälöä

$$x'(t) = ax(t) + b(t), \quad \text{tai yhtäpitävästi} \quad x'(t) - ax(t) = b(t). \tag{2.21}$$

Epähomogeenisen yhtälön yleinen ratkaisu saadaan homogeenisen yhtälön ($b(t) = 0$) yleisen ratkaisun $x_h(t)$ sekä epähomogeenisen yhtälön **yksityisratkaisun** $x_p(t)$ summana. Aiemmasta tiedetään, että $x_h(t) = \mathrm{e}^{at}c$. Lisäksi voidaan osoittaa, että

$$x_p(t) = \mathrm{e}^{at} \int \mathrm{e}^{-as} b(s) \mathrm{d}s,$$

jolloin yleiseksi ratkaisuksi saadaan

$$x(t) = x_h(t) + x_p(t) = \mathrm{e}^{at}c + \mathrm{e}^{at} \int \mathrm{e}^{-at} b(t) \mathrm{d}t. \tag{2.22}$$

Alkuehdolla $x(0) = x_0 \in \mathbb{R}$ epähomogeenisen yhtälön ratkaisuksi saadaan

$$x(t) = \mathrm{e}^{at}x_0 + \mathrm{e}^{at} \int_0^t \mathrm{e}^{-as} b(s) \mathrm{d}s = \mathrm{e}^{at}x_0 + \int_0^t \mathrm{e}^{a(t-s)} b(s) \mathrm{d}s. \tag{2.23}$$

2.2.1 Lineaariset systeemit

Tarkastellaan seuraavaksi 1. kertaluvun lineaarisia ja vakiokertoimisia **differentiaaliyhtälösysteemejä**. Ne ovat muotoa

$$\begin{cases} x_1'(t) = a_{11}x_1(t) + a_{12}x_2(t) + \ldots + a_{1n}x_n(t) \\ x_2'(t) = a_{21}x_1(t) + a_{22}x_2(t) + \ldots + a_{2n}x_n(t) \\ \quad \vdots \\ x_n'(t) = a_{n1}x_1(t) + a_{n2}x_2(t) + \ldots + a_{nn}x_n(t) \end{cases},$$

tai matriisimerkinnöillä

$$\mathbf{x}'(t) = A\mathbf{x}(t). \tag{2.24}$$

Haettavana on joko yleinen ratkaisu alkuehdon $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^n$ toteuttava ratkaisu. Vektoria $\mathbf{x}(t) \in \mathbb{R}^n$ sanotaan systeemin **tilaksi**. Nyt A on kokoa $n \times n$ oleva vakiomatriisi.

Olkoon yhtälössä (2.24) esiintyvä A reaalinen ja diagonalisoituva matriisi. Tällöin Lauseen 2.1.8 nojalla tiedetään, että on olemassa ei-singulaarinen matriisi $Q = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$ siten, että $A = QDQ^{-1}$, missä vektorit $\mathbf{v}_i$ ovat A:n lineaarisesti riippumattomia ominaisvektoreita, ja diagonaalimatriisin $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ lävistäjällä on A:n ominaisarvot λ_i, kun $i = 1, 2, \ldots, n$.

Lause 2.2.1. *Olkoon A diagonalisoituva matriisi siten, että $A = QDQ^{-1}$. Tällöin*

$$A^m = QD^mQ^{-1}, \quad m \in \mathbb{N}.$$

Todistus. Käytetään induktiota luvun m suhteen. Tapaus $m = 1$ on selvä. Oletetaan, että $A^l = QD^lQ^{-1}$. Tällöin

$$A^{l+1} = A^lA = (QD^lQ^{-1})(QDQ^{-1}) = QD^lDQ^{-1} = QD^{l+1}Q^{-1}.$$

Siis väite pätee kaikilla $m \in \mathbb{N}$. □

Nyt sarjakehitelmän $\mathrm{e}^A = \sum_{i=0}^{\infty} \frac{A^i}{i!}$ sekä Lauseen 2.2.1 avulla saadaan seuraava tulos:

Lause 2.2.2. *Olkoon A diagonalisoituva $n \times n$ -matriisi siten, että $A = QDQ^{-1}$. Tällöin*

$$\mathrm{e}^A = Q\mathrm{e}^DQ^{-1}, \tag{2.25}$$

missä $\mathrm{e}^D = \operatorname{diag}(\mathrm{e}^{\lambda_1}, \mathrm{e}^{\lambda_2}, \ldots, \mathrm{e}^{\lambda_n})$.

Todistus. Sivuutetaan. □

Lauseen 2.2.2 avulla voidaan muodostaa seuraava matriisieksponenttifunktio:

$$\mathrm{e}^{tA} = Q\mathrm{e}^{tD}Q^{-1}, \tag{2.26}$$

tai aukikirjoitettuna

$$\mathrm{e}^{tA} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \mathrm{e}^{\lambda_1 t} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{\lambda_2 t} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{\lambda_n t} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}^{-1},$$

missä λ_i on A:n ominaisarvo ja $\mathbf{v}_i$ sitä vastaava ominaisvektori, $i = 1, 2, \dots, n$.

Lause 2.2.3. *Olkoon A $n \times n$ -matriisi, $\mathbf{x}(t) \in \mathbb{R}^n$ ja $\mathbf{x}_0 \in \mathbb{R}^n$. Alkuarvoprobleeman*

$$\mathbf{x}'(t) = A\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.27}$$

ratkaisuna on vektorifunktio

$$\mathbf{x}(t) = \mathrm{e}^{tA}\mathbf{x}_0. \tag{2.28}$$

Todistus. Alkuehto toteutuu, sillä $\mathbf{x}(0) = \mathrm{e}^{0A}\mathbf{x}_0 \overset{\mathrm{e}^{0A}=I}{=} \mathbf{x}_0$. Viitteen [9, s. 128] nojalla $\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{e}^{tA} = A\mathrm{e}^{tA}$. Tällöin suoraan derivoimalla saadaan, että

$$\mathbf{x}'(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \frac{\mathrm{d}}{\mathrm{d}t}(\mathrm{e}^{tA})\mathbf{x}_0 = A\mathrm{e}^{tA}\mathbf{x}_0 = A\mathbf{x}(t).$$

Siis vektorifunktio (2.28) toteuttaa alkuarvoprobleeman (2.27). □

Jos A on diagonalisoituva, matriisieksponentti e^{tA} voidaan laskea kaavan (2.26) avulla. Muussa tapauksessa e^{tA} voidaan muodostaa esimerkiksi viitteessä [18] mainituilla numeerisilla menetelmillä. Lause 2.2.3 on joka tapauksessa voimassa, vaikkei A olisikaan diagonalisoituva.

Tarkastellaan seuraavaksi epähomogeenisen differentiaaliyhtälösysteemin alkuarvoprobleemaa

$$\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}(t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2.29}$$

Oletetaan edelleen, että A on kokoa $n \times n$ oleva vakiomatriisi, $\mathbf{x}(t)$ ja $\mathbf{x}_0$ ovat $n \times 1$ -vektoreita ja $\mathbf{b}(t)$ on jatkuva vektorifunktio. Kuten yksiulotteisessa tapauksessa, epähomogeenisen yhtälön ratkaisu on muotoa

$$\mathbf{x}(t) = \mathbf{x}_h(t) + \mathbf{x}_p(t),$$

missä $\mathbf{x}_h(t)$ on homogeenisen systeemin $\mathbf{x}'(t) = A\mathbf{x}(t)$ yleinen ratkaisu ja $\mathbf{x}_p(t)$ on epähomogeenisen systeemin $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}(t)$ jokin yksityisratkaisu. Yksiulotteisessa tapauksessa ratkaisuksi saatiin $x(t) = \mathrm{e}^{at}x_0 + \int_0^t \mathrm{e}^{a(t-s)}b(s)\mathrm{d}s$. Osoittautuu, että differentiaaliyhtälösysteemeillä ratkaisu on samanlaista muotoa:

$$\mathbf{x}(t) = \mathrm{e}^{tA}\mathbf{x}_0 + \int_0^t \mathrm{e}^{(t-s)A}\mathbf{b}(s)\mathrm{d}s. \tag{2.30}$$

Suoraan derivoimalla voidaan vakuuttua siitä, että (2.30) todellakin toteuttaa alkuarvoprobleeman (2.29).

Lause 2.2.4. *Olkoon A ei-singulaarinen $n \times n$ -matriisi ja $\mathbf{b}(t) \equiv \mathbf{b}$ vakiovektori. Tällöin alkuarvoprobleeman* (2.29) *ratkaisu on*

$$\mathbf{x}(t) = \mathrm{e}^{tA}(\mathbf{x}_0 + A^{-1}\mathbf{b}) - A^{-1}\mathbf{b}. \tag{2.31}$$

Todistus. Alkuehto toteutuu, sillä $\mathbf{x}(0) = \mathbf{x}_0 + A^{-1}\mathbf{b} - A^{-1}\mathbf{b} = \mathbf{x}_0$, ja

$$\mathbf{x}'(t) = A\underbrace{\mathrm{e}^{tA}(\mathbf{x}_0 + A^{-1}\mathbf{b})}_{=\mathbf{x}(t)+A^{-1}\mathbf{b}} = A\mathbf{x}(t) + \mathbf{b}.$$

□

2.2.2 Tasapainotilat ja stabiilius

Tässä kappaleessa tarkastellaan differentiaaliyhtälösysteemejä ja esitellään stabiiliuden käsite. Seuraavissa määritelmissä vektori $\mathbf{x}(t) \in \mathbb{R}^n$ on systeemin tila ajanhetkellä t, ja $\mathbf{f}(t, \mathbf{x}(t))$ on jatkuva funktio, $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$.

Määritelmä 2.2.5. Differentiaaliyhtälön $\mathbf{x}'(t) = \mathbf{f}(t, \mathbf{x}(t))$ määrittelemä systeemi on **autonominen**, jos yhtälön oikea puoli ei riipu eksplisiittisesti ajasta t. Toisin sanoen, $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$. ■

Määritelmä 2.2.6. Jos vakiotila $\mathbf{x}(t) = \hat{\mathbf{x}}$ toteuttaa yhtälön $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$, niin tilan derivaatta on vakiona $\mathbf{x}'(t) = \mathbf{0}$. Tällöin sanotaan, että systeemi on **tasapainotilassa** (*engl. steady state*) ja $\hat{\mathbf{x}}$ on systeemin **tasapainopiste** (*engl. equilibrium point*). Systeemin tasapainopisteet voidaan ratkaista yhtälöstä

$$\mathbf{f}(\hat{\mathbf{x}}) = \mathbf{0}. \tag{2.32}$$

■

Määritelmä 2.2.7. Systeemi $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$ on tasapainopisteessä $\hat{\mathbf{x}}$ **asymptoottisesti stabiili**, jos

$$\lim_{t\to\infty} \mathbf{x}(t) = \hat{\mathbf{x}}.$$

Toisin sanottuna, ajan kuluessa systeemi palaa takaisin tasapainotilaansa. ■

Mikäli systeemi ei ole stabiili, se on **epästabiili**. Tällöin systeemin tila ei pysy edes kiinteän etäisyyden päässä tasapainopisteestä vaan karkaa siitä poispäin.

Tarkastellaan seuraavaksi lineaarista differentiaaliyhtälösysteemiä $\mathbf{x}'(t) = A\mathbf{x}(t)$. Oletetaan, että $\det(A) \neq 0$. Määritelmän 2.2.6 nojalla systeemin tasapainopisteet voidaan ratkaista yhtälöstä $A\hat{\mathbf{x}} = \mathbf{0}$. Koska A on ei-singulaarinen, systeemin ainoa tasapainopiste on origo $\mathbf{0}$. Voidaan osoittaa, että origo on systeemin asymptoottisesti stabiili tasapainotila, jos A:n kaikkien ominaisarvojen reaaliosat ovat pienempiä kuin 0.

Selvitetään vielä epähomogeenisen systeemin tasapainopisteet. Tarkastellaan systeemiä $\mathbf{x}'(t) = A\mathbf{x}(t)+\mathbf{b}$, missä $\mathbf{b}$ on vakiovektori, ja A on ei-singulaarinen vakiomatriisi. Tasapainopisteet saadaan yhtälöstä $A\hat{\mathbf{x}} + \mathbf{b} = \mathbf{0}$. Koska A on ei-singulaarinen, yhtälöllä on yksikäsitteinen ratkaisu, jolloin ainoana tasapainopisteenä on

$$\hat{\mathbf{x}} = -A^{-1}\mathbf{b}. \tag{2.33}$$

Kootaan epähomogeenisen systeemin stabiiliusehdot seuraavaan lauseeseen, jonka todistus sivuutetaan.

Lause 2.2.8. *Olkoon A ei-singulaarinen vakiomatriisi. Lineaarisella systeemillä*

$$\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}$$

tasapainotila $\hat{\mathbf{x}} = -A^{-1}\mathbf{b}$ on asymptoottisesti stabiili, jos A:n kaikki ominaisarvot $\lambda \in \mathbb{C}$ sijaitsevat aidosti vasemmassa puolitasossa. Toisin sanottuna, $\mathrm{Re}(\lambda) < 0$. *Sen sijaan jos yksikin A:n ominaisarvoista on reaaliosaltaan positiivinen, systeemi on tasapainotilassaan epästabiili.*

2.2.3 Numeerisista ratkaisijoista

Tarkastellaan systeemin (2.29) ratkaisua tietokonelaskennan avulla, kun käytetään MATLAB-ohjelmistoa. Tiedetään, että ratkaisufunktiossa esiintyy matriisieksponentti e^{tA}. Se voidaan laskea käyttämällä MATLAB-funktiota `expm`, joka käyttää **Padé-approksimaatiota** sekä **scaling and squaring** -menetelmää (viite [18], menetelmä 3). Matriisieksponentin muodostaminen on kuitenkin laskennallisesti epätarkkaa, ja sopivan aika-askeleen $\Delta t = t_2 - t_1$ valinta ratkaisujen $\mathbf{x}(t_1)$ ja $\mathbf{x}(t_2)$

laskemiseksi tuo myös lisähaastetta. Käytetään siksi **numeerisia ratkaisijoita**. MATLAB:issa on seitsemän erilaista numeerista ratkaisijafunktiota, joissa kaikissa aika-askeleen valinta on toteutettu algoritmin sisällä.

Tyypillisesti differentiaaliyhtälöiden ratkaisuun käytetään ratkaisijoita `ode45`, `ode23` ja `ode113`. Niin sanotulle **jäykälle** (*engl. stiff*) differentiaaliyhtälösysteemille ei ole olemassa virallista määritelmää, mutta seuraavaa voidaan pitää suuntaa antavana:

Määritelmä 2.2.9. Olkoon A kokoa $n \times n$ oleva vakiomatriisi systeemissä $\mathbf{x}'(t) = A\mathbf{x}(t) + \mathbf{b}(t)$ ja λ_1, λ_2, …, λ_n A:n ominaisarvot. Jos suhde

$$\frac{\max\limits_{i \in I} |\mathrm{Re}(\lambda_i)|}{\min\limits_{i \in I} |\mathrm{Re}(\lambda_i)|}, \quad I = \{1, 2, \ldots, n\}$$

on suuri, systeemiä voidaan pitää jäykkänä. ■

Jäykillä systeemeillä ratkaisu sisältää tyypillisesti komponentteja, jotka muuttuvat hyvin nopeasti ajan suhteen. Nämä muutokset vaikeuttavat ja hidastavat differentiaaliyhtälön ratkaisua. Jäykkiä systeemejä varten MATLAB:illa on käytössä erityisiä ratkaisijoita: `ode15s`, `ode23s` (`s` tulee sanasta *stiff*), `ode23t` ja `ode23tb`. Tavallisesti pyritään kuitenkin välttämään jäykkiä systeemejä, jos mahdollista, ja tässä opinnäytetyössä yritetäänkin saattaa systeemi sellaiseen muotoon, että ominaisarvojen reaaliosat eivät suuruusluokaltaan poikkea toisistaan kovin paljon. Tällöin systeemi ei ole jäykkä.

Esitellään MATLAB:in ratkaisijafunktiot vielä yksitellen.

1. `ode45` [6]: yleisin ratkaisijafunktio, jota tyypillisesti kokeillaan ensimmäisenä differentiaaliyhtälöiden ratkaisemiseksi. Funktio perustuu ekspliittiseen, 4. ja 5. kertaluvun Runge–Kutta-kaavaan, Dormand–Prince-menetelmään. Kyseessä on yhden askeleen ratkaisija. Toisin sanottuna, ratkaisun $\mathbf{x}(t_i)$ laskemiseksi tarvitsee vain tietää ratkaisu edellisellä ajanhetkellä, $\mathbf{x}(t_{i+1})$.

2. `ode23` [3]: käyttää eksplisiittistä, 2. ja 3. kertaluvun Runge–Kuttaa, Bogacki–Shampine-menetelmää. Kuten `ode45`, myös `ode23` on yhden askeleen ratkaisija. Korkeamman kertaluvun versioonsa nähden `ode23` on hyödyllinen sellaisissa tilanteissa, joissa tarvitaan karkeita toleransseja, tai kun systeemissä on mukana lievää jäykkyyttä.

3. `ode113` [24]: vaihtelevan kertaluvun Adams–Bashforth–Moulton-menetelmää käyttävä PECE-ratkaisija. Funktion on hyödyllinen tilanteissa, joissa tarvitaan tiukkoja toleransseja. Toisin kuin kaksi edellistä, `ode113` on usean askeleen ratkaisija. Se siis tarvitsee ratkaisut usealla edellisellä ajanhetkellä, jotta

ratkaisu voidaan laskea uudella ajanhetkellä.

4. `ode15s` [26]: vaihtelevan kertaluvun ratkaisija, jonka toiminta perustuu numeeristen derivointikaavojen (*engl. numerical differentiation formulas, lyh. NDFs*) käyttöön. Vaihtoehtoisesti funktio voi käyttää taaksepäin derivoinnin kaavoja (*engl. backward differentiation formulas, lyh. BDFs*). Aivan kuten `ode113`, myös `ode15s` on usean askeleen ratkaisija. Se on hyödyllinen jäykkien systeemien tapauksissa ja differentiaali-algebrallisten yhtälöiden (*engl. Differential-Algebraic Equations, lyh. DAE*, [27]) ratkaisemisessa. Mikäli `ode45` ei pysty ratkaisemaan differentiaaliyhtälöä tai on liian hidas, tavallisesti `ode15s`:ää kokeillaan seuraavaksi.

5. `ode23s` [26]: perustuu 2. kertaluvun muunnettuun Rosenbrockin kaavaan. Kuten tavallinen `ode23`, myös `ode23s` on yhden askeleen ratkaisija. Se on tarkoitettu jäykille systeemeille, ja `ode15s`:ään verrattuna se on hyödyllinen tilanteissa, joissa tarvitaan karkeita toleransseja.

6. `ode23t` [27]: käyttää puolisuunnikassääntöä (*engl. trapezoidal rule*). Funktio on hyödyllinen tilanteissa, joissa systeemi on ainoastaan kohtuullisen jäykkä, tai jos ratkaisuun ei haluta numeerista vaimenemista. Ratkaisijan `ode15s` tapaan `ode23` voi ratkaista differentiaali-algebrallisia yhtälöitä.

7. `ode23tb` [25]: käyttää TR–BDF2-menetelmää, joka perustuu implisiittiseen Runge–Kutta-kaavaan. Funktiossa on kaksi vaihetta. Ensimmäisessä vaiheessa otetaan askel puolisuunnikassäännöllä, ja toisessa vaiheessa käytetään 2. kertaluvun taaksepäin derivoinnin kaavoja [2]. Kummankin vaiheen iteraatioissa käytetään samaa matriisia. Funktio on tarkoitettu jäykille systeemeille, ja se on hyödyllinen tilanteissa, joissa tarvitaan karkeita toleransseja.

Jatkossa käytetään lähinnä `ode45`-ratkaisijafunktiota, mutta jos systeemi on jäykkä, käytetään `ode15s`:ää.

2.3 Systeemiteoriaa

Tässä kappaleessa esitellään muutama systeemiteoriaan liittyvä peruskäsite. Systeemiteoriasta voi lukea tarkemmin alan perusteoksista, esimerkiksi [11] ja [12]. Tämän luvun tarkastelu on lähinnä peräisin viitteistä [15], [29] ja [30].

Määritelmä 2.3.1. Olkoon A $n \times n$-matriisi, B $n \times p$-matriisi, C $p \times n$-matriisi ja

D $p \times p$-matriisi, $p \leq n$. Tarkastellaan yhtälöryhmää

$$\begin{cases} \mathbf{x}'(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t) \\ \mathbf{x}(0) = \mathbf{x}_0, \quad t \geq 0 \end{cases}. \tag{2.34}$$

Systeemiä (2.34) sanotaan **tila-avaruussysteemiksi** (*engl. state space system*). Vektori $\mathbf{x}(t)$ on **tila** (*engl. state*), $\mathbf{u}(t)$ on **sisääntulo** (*engl. input*) tai **ohjaus** (*engl. control*) ja $\mathbf{y}(t)$ on **ulostulo** (*engl. output*). Matriiseista käytetään seuraavia nimityksiä: A on **tilamatriisi**, B on **sisääntulomatriisi**, C on **ulostulomatriisi** ja D on **myötäkytkentämatriisi** (engl. feedforward matrix). **Aika** $t \in \mathbb{R}$ oletetaan tässä opinnäytetyössä jatkuvaksi. ■

Esitetään seuraavaksi muutama määritelmä ja lause, joita ei todisteta.

Määritelmä 2.3.2. Systeemin (2.34) **ohjattavuus** (*engl. controllability*) tarkoittaa sitä, että tila $\mathbf{x}(t)$ voidaan sisääntulon $\mathbf{u}(t)$ avulla ohjata mielivaltaisesta alkutilasta $\mathbf{x}_0$ mihin tahansa lopputilaan. ■

Lause 2.3.3. *Määritellään systeemin* (2.34) ***ohjattavuusmatriisi***

$$\mathcal{C} = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix}.$$

Systeemi on ohjattavissa, jos ja vain jos $\operatorname{rank}(\mathcal{C}) = n$.

Määritelmä 2.3.4. Systeemin (2.34) **tarkkailtavuus** (*engl. observability*) tarkoittaa sitä, että systeemin alkutila voidaan selvittää ulostulon $\mathbf{y}(t)$ perusteella. ■

Lause 2.3.5. *Määritellään systeemin* (2.34) ***tarkkailtavuusmatriisi***

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

Systeemi on tarkkailtavissa, jos ja vain jos $\operatorname{rank}(\mathcal{O}) = n$.

Ohjattavuus ja tarkkailtavuus ovat keskenään käsitepari siinä mielessä, että ohjattavuudessa saadaan sisääntulon avulla vietyä alkutila haluttuun lopputilaan, ja tarkkailtavuudessa saadaan ulostulon avulla selvitettyä alkutila.

Määritelmä 2.3.6. Systeemi (2.34) on **stabiloitavissa**, jos on olemassa sellainen **takaisinkytkentä** (*engl. feedback*) $\mathbf{u}(t) = F\mathbf{x}(t)$, että systeemi saadaan stabiiliksi (toisin sanottuna, matriisin $A + BF$ ominaisarvot saadaan asetettua avoimeen vasempaan puolitasoon). ■

Systeemin ohjattavuudesta seuraa aina sen stabiloitavuus. Takaisinkytkennässä $\mathbf{u}(t) = F\mathbf{x}(t)$ käytetty matriisi F voidaan muodostaa esimerkiksi MATLAB:in funktioilla `lqr` ja `place`. Näistä jälkimmäinen sijoittaa matriisin $A + BF$ ominaisarvot haluttuihin pisteisiin kompleksitasossa [14]. Kun F on saatu jommalla kummalla menetelmällä muodostettua, matriisin $A + BF$ ominaisarvot sijaitsevat avoimessa vasemmassa puolitasossa. Matriisin F muodostaminen voi kuitenkin olla laskennallisesti työlästä, joten jatkossa pyritään valitsemaan systeemin (2.34) lohkot siten, että systeemi on jo valmiiksi stabiili, jopa ilman takaisinkytkentää. Lohkojen valintaan palataan kappaleessa 3.2.

3. DY-RATKAISIJAN TOIMINTA JA PARAMETRIT

3.1 Tutkimuskysymys

Opinnäytetyön tavoitteena on etsiä laskennallisesta näkökulmasta mahdollisimman nopea menetelmä, jolla voidaan ratkaista N kappaletta lineaarisia yhtälöryhmiä

$$C_i \mathbf{x}_i = \mathbf{r}_i, \quad i = 1, 2, \ldots, N, \tag{3.1}$$

missä C_i on $p \times n$ -matriisi ($p \leq n$), $\mathbf{x}_i$ on tuntematon $n \times 1$ -vektori ja $\mathbf{r}_i$ on $p \times 1$ -vektori. Halutaan siis toistuvasti ratkaista tuntematon vektori $\mathbf{x}_i$ yhtälöryhmästä (3.1). Asetetaan probleemalle sellainen oletus, että peräkkäiset matriisit C_i sekä vektorit $\mathbf{r}_i$ ovat hyvin lähellä toisiaan, eli

$$C_{i+1} \approx C_i, \qquad \mathbf{r}_{i+1} \approx \mathbf{r}_i.$$

Lähestytään probleemaa seuraavan periaatteen mukaisesti. Ensin ratkaistaan yhtälöryhmä $C_i \mathbf{x}_i = \mathbf{r}_i$. Tämän jälkeen kohdistetaan matriisiin C ja vektoriin $\mathbf{r}$ pieni häiriö:

$$C_{i+1} = C_i + \Delta C_i, \qquad \mathbf{r}_{i+1} = \mathbf{r}_i + \Delta \mathbf{r}_i,$$

missä häiriöt ΔC_i ja $\Delta \mathbf{r}_i$ ovat pieniä. Tässä pienillä häiriöillä tarkoitetaan sitä, että matriisin ΔC_i ja vektorin $\mathbf{r}_i$ alkiot ovat pieniä. Häiriö voi kohdistua myös pelkkään matriisiin C_i tai vektoriin $\mathbf{r}_i$. Toisin sanottuna, saattaa olla $\Delta C_i = O$ tai $\Delta \mathbf{r}_i = \mathbf{0}$. Tutkimuskysymyksen kannalta on olennaista keskittyä matriisin C_i häiriöihin. Vektorin $\mathbf{r}_i$ häiriöiden käsittely osoittautuu melko triviaaliksi.

Lineaariset yhtälöryhmät ratkaistaan tyypillisesti kappaleessa 2.1.1 käsitellyn **LU-hajotelman** avulla. Menetelmä on yleisesti hyvin tunnettu ja laajalti käytetty tietokonelaskennassa. Algoritmi on melko hyvin optimoitu erityyppisille matriiseille, ja lineaaristen yhtälöryhmien ratkaisu on nopeaa. LU-hajotelman heikkoutena on se, että kerroinmatriisin C_i muuttuessa jokainen yhtälöryhmä $C_i \mathbf{x}_i = \mathbf{r}_i$ on ratkaistava erillisenä tehtävänä, jolloin yhtälö on ratkaistava N kertaa. Jos yksittäisen yhtälöryhmän $C_i \mathbf{x}_i = \mathbf{r}_i$ ratkaisemiseen vaaditaan $\frac{4}{3}n^3$ flopsia, probleeman (3.1) ratkaisemiseen vaaditaan $N \cdot \frac{4}{3}n^3$ flopsia.

Tarkastellaan toista tapaa ratkaista probleema (3.1). Muodostetaan asymptoottisesti stabiili differentiaaliyhtälösysteemi, joka tasapainotilassa lähestyy sellaista pistettä $\hat{\mathbf{x}}_i$, joka toteuttaa yhtälön $C_i\mathbf{x}_i = \mathbf{r}_i$. Toisin sanottuna,

$$\lim_{t\to\infty} \mathbf{x}(t) = \hat{\mathbf{x}}_i \qquad \text{ja} \qquad C_i\hat{\mathbf{x}}_i = \mathbf{r}_i.$$

Tavoitteena on siis etsiä tämä tasapainotila $\hat{\mathbf{x}}_i$. Arvion (2.17) perusteella tiedetään, että hyvin pienillä C_i:n häiriöillä kahden peräkkäisen systeemin ratkaisut ovat hyvin lähellä toisiaan, eli $\mathbf{x}_{i+1} \approx \mathbf{x}_i$. Tällöin myös tasapainotiloille pätee, että $\hat{\mathbf{x}}_{i+1} \approx \hat{\mathbf{x}}_i$. Jos yhtälössä $C_{i+1}\mathbf{x}_{i+1} = \mathbf{r}_{i+1}$ edellinen ratkaisu $\mathbf{x}_i$ asetetaan häirityn systeemin alkutilaksi, uuden ratkaisun $\mathbf{x}_{i+1}$ selvittäminen on laskennallisesti nopeaa, sillä alkutila ja uusi tasapainotila ovat hyvin lähellä toisiaan. Menetelmä perustuu **robustiin säätötekniikkaan** [5]. Koska menetelmää ei ole yleisesti toteutettu tietokonelaskennassa, tässä opinnäytetyössä pyritään ensin muodostamaan mahdollisimman yksinkertainen differentiaaliyhtälösysteemi, jonka tasapainotilana on yhtälön $C_i\mathbf{x}_i = \mathbf{r}_i$ ratkaisu. Kun systeemi on saatu rakennettua, menetelmään liittyviä parametreja pyritään vielä optimoimaan, jotta ratkaisija toimisi mahdollisimman nopeasti. Kuvattua menetelmää kutsutaan **DY-ratkaisijaksi**.

Luvussa 3 keskitytään DY-ratkaisijan kehittämiseen seuraavan strategian mukaisesti:

1. Muodostetaan differentiaaliyhtälösysteemi.
2. Valitaan differentiaaliyhtälön matriisit siten, että systeemi on stabiili ja tasapainotilana on yhtälön $C_i\mathbf{x}_i = \mathbf{r}_i$ (jokin) ratkaisu.
3. Simuloidaan differentiaaliyhtälöä numeerisesti, jotta saataisiin approksimaatio tasapainotilasta.

Luvussa 4 on kuvattu algoritmi, jolla sekä LU-hajotelman että DY-ratkaisijan nopeutta ratkaista probleema (3.1) verrataan keskenään. Tämän jälkeen menetelmiä on verrattu eri kokoisilla matriiseilla. Tavoitteena on selvittää, kumpi menetelmä on nopeampi ratkaisemaan probleemaa (3.1).

3.2 DY-ratkaisijan toimintaperiaate

Tässä kappaleessa esitellään DY-ratkaisijan yleinen toimintaperiaate teoreettiselta kannalta. Ideana on, että yhtälöryhmä

$$C\mathbf{x} = \mathbf{r} \tag{3.2}$$

ratkaistaan lineaarisen differentiaaliyhtälösysteemin avulla. Nyt C on $p \times n$ -kerroinmatriisi ($p \leq n$), $\mathbf{r}$ on jokin annettu vektori ja $\mathbf{x}$ on yhtälöryhmän ratkaisu.

Tarkastellaan yhtälöryhmää (2.34) muistuttavaa systeemiä

$$\mathbf{x}_c'(t) = A_c\mathbf{x}_c(t) + \mathbf{r}_c, \quad \mathbf{x}_c(0) = \mathbf{x}_{c,0}, \quad t \geq 0, \tag{3.3a}$$

missä

$$\mathbf{x}_c(t) = \begin{bmatrix} \mathbf{x}(t) \\ \xi(t) \end{bmatrix}, \quad A_c = \begin{bmatrix} A & B \\ C & O \end{bmatrix} \quad \text{ja} \quad \mathbf{r}_c = \begin{bmatrix} \mathbf{0} \\ -\mathbf{r} \end{bmatrix}. \tag{3.3b}$$

Nyt siis tilavektorina on $\mathbf{x}_c(t)$, tilamatriisina A_c, sisääntulona vakiovektori $\mathbf{r}_c$ ja ulostuloa ei ole. Matriisi A on kokoa $n \times n$, B kokoa $n \times p$ ja C kokoa $p \times n$, ja lisäksi $p \leq n$. Vektoreille on voimassa $\mathbf{x} \in \mathbb{R}^n$, $\xi \in \mathbb{R}^p$ ja $\mathbf{r} \in \mathbb{R}^p$. Matriisi A_c on siis kokoa $(n+p) \times (n+p)$, ja vektorien $\mathbf{x}_c$ ja $\mathbf{r}_c$ koko on $(n+p) \times 1$. Luvusta 2.2.2 tiedetään, että systeemi (3.3a) on asymptoottisesti stabiili, jos matriisin A_c ominaisarvot λ_1, λ_2, …, λ_{n+p} sijaitsevat kompleksitason avoimessa vasemmassa puolitasossa, eli

$$\mathrm{Re}(\lambda_i) < 0 \quad \forall i = 1, 2, \ldots, n+p. \tag{3.4}$$

Ehdosta (3.4) seuraa, että mikään A_c:n ominaisarvo ei voi olla 0. Tämä on yhtäpitävää sen kanssa, että stabiilin systeemin tapauksessa matriisin A_c on oltava ei-singulaarinen.

Stabiili systeemi lähestyy ajan kuluessa tasapainotilaansa. Määritelmän 2.2.6 nojalla tasapainotilassa $\mathbf{x}_c'(t) = \mathbf{0}$, jolloin tasapainopisteet voidaan ratkaista yhtälöstä (2.32):

$$\begin{bmatrix} A & B \\ C & O \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\xi} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Matriisiyhtälön alemmalta riviltä saadaan, että $C\hat{\mathbf{x}} = \mathbf{r}$. Siis ajan mennessä ääretömään tila $\mathbf{x}(t)$ lähestyy lineaarisen yhtälöryhmän (3.2) ratkaisua.

Stabiilin systeemin tapauksessa lineaarinen yhtälöryhmä (3.2) voidaan siis ratkaista hakemalla lineaarisen differentiaaliyhtälösysteemin (3.3a) ratkaisu, jolloin $\hat{\mathbf{x}} = \lim_{t\to\infty} \mathbf{x}(t)$ toteuttaa yhtälöryhmän. Huomaa, että tasapainotila saavutetaan vain stabiilin systeemin tapauksessa kaikilla alkutiloilla. Mikäli systeemi onkin epästabiili, eli jokin matriisin A_c ominaisarvoista sijaitsee kompleksitason suljetussa oikeassa puolitasossa, tila $\mathbf{x}(t)$ ei välttämättä lähesty yhtälöryhmän (3.2) ratkaisua. Oletetaan, että matriisilla A_c on $(k+1)$-kertainen ominaisarvo $\lambda = \alpha + \mathrm{i}\beta$, missä $\alpha \geq 0$. Tällöin ratkaisun $\mathbf{x}_c(t)$ komponenteista osa on muotoa

$$t^k \mathrm{e}^{(\alpha+\mathrm{i}\beta)t} = t^k \mathrm{e}^{\alpha t} \left(\cos(\beta t) + \mathrm{i}\sin(\beta t)\right),$$

eikä ratkaisu yleensä suppene. Ajan kuluessa systeemi etenee poispäin tasapainotilastaan joillain alkutiloilla.

Laskennallisesti riittää, että tulo $C\mathbf{x}(t)$ on riittävän lähellä vektoria $\mathbf{r}$. Muodos-

tetaan lauseke, joka kuvaa sitä, kuinka paljon $C\mathbf{x}$ ja $\mathbf{r}$ poikkeavat toisistaan:

$$\epsilon(t) = \frac{\|\mathbf{r} - C\mathbf{x}(t)\|}{\|\mathbf{r}\|}. \tag{3.5}$$

Funktio $\epsilon(t)$ kuvaa siis yhtälön $C\mathbf{x}(t) = \mathbf{r}$ suhteellista virhettä ajanhetkellä t. Kun jostain ajanhetkestä t^* lähtien virhe $\epsilon(t)$ pysyy jotain valittavissa olevaa toleranssirajaa ϵ_{tol} pienempänä, voidaan hyväksyä, että systeemin tila $\mathbf{x}_c(t)$ on riittävän lähellä tasapainotilaa $\hat{\mathbf{x}}$. Tällöin jokainen ratkaisu $\mathbf{x}(t)$, jolle $\epsilon(t) < \epsilon_{\text{tol}}$, kelpaa lineaarisen yhtälöryhmän ratkaisuksi. Valitaan siis ratkaisuksi ensimmäinen havaittu $\mathbf{x}(t)$, jolle on voimassa

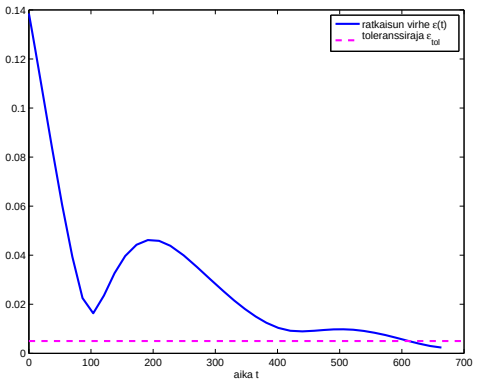
$$\epsilon(t) = \frac{\|\mathbf{r} - C\mathbf{x}(t)\|}{\|\mathbf{r}\|} < \epsilon_{\text{tol}}. \tag{3.6}$$

Virheen $\epsilon(t)$ käyttäytymistä ajan funktiona on havainnollistettu kuvassa 3.1. Lopulta, jos systeemi on stabiili, virhe menee pienemmäksi kuin ϵ_{tol}. Kuvassa 3.1 on valittu toleranssiksi $\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}$, ja tätä arvoa käytetään jatkossakin.

Määritelmä 3.2.1. Yhtälössä (3.6) mainituksi toleranssiksi valitaan oletusarvoisesti

$$\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}, \tag{3.7}$$

ellei toisin mainita. ■



Kuva 3.1: Kuvaaja yhtälön $C\mathbf{x}(t) = \mathbf{r}$ suhteellisesta virheestä $\epsilon(t)$ ajan funktiona. Lopulta virhe pysyy tiettyä toleranssia pienempänä. Tällöin mikä tahansa tila $\mathbf{x}(t)$, jota vastaava virhe $\epsilon(t)$ pysyy pienempänä kuin ϵ_{tol}, voidaan hyväksiä lineaarisen yhtälöryhmän $C\mathbf{x} = \mathbf{r}$ ratkaisuksi. Tässä kuvassa ja jatkossa käytetään toleranssia $\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}$.

Systeemi voidaan todeta epästabiiliksi laskemalla matriisin A_c ominaisarvot. Jos jokin ominaisarvo sijaitsee kompleksitason suljetussa oikeassa puolitasossa, systeemi on epästabiili. Ominaisarvojen selvittäminen on kuitenkin laskennallisesti työlästä,

varsinkin suurille matriiseille. Vaihtoehtoisesti systeemi voidaan laskennallisesti todeta epästabiiliksi, jos jollain ajanhetkellä t' virhe $\epsilon(t)$ kasvaa liian suureksi:

$$\epsilon(t') = \frac{\|\mathbf{r} - C\mathbf{x}(t')\|}{\|\mathbf{r}\|} > M, \tag{3.8}$$

missä $M > 0$ on valittavissa oleva suuri luku, esimerkiksi $M = 1{,}0 \cdot 10^6$. Virheen kasvaminen suureksi ei tietenkään aina todista systeemiä epästabiiliksi, mutta riittävän suuri virhe antaa useimmiten aihetta epäillä, että virhe karkaisi äärettömään eikä tasapainotilaa saavutettaisi.

Tarkastellaan systeemiä (3.3a), ja oletetaan matriisi C ja vektori $\mathbf{r}$ tunnetuiksi. Matriisit A ja B voidaan vapaasti valita, joten valitaan ne siten, että systeemi (3.3a) olisi asymptoottisesti stabiili. Toisin sanottuna, valitaan A ja B siten, että matriisin A_c kaikki ominaisarvot ovat reaaliosaltaan aidosti negatiivisia. Pyritään siihen, että systeemi olisi mahdollisimman yksinkertainen. Tarkastellaan ensin 2×2 -matriisia

$$A_c = \begin{bmatrix} a & b \\ c & 0 \end{bmatrix},$$

ja oletetaan, että $a \neq 0$, $b \neq 0$ ja $c \neq 0$. Matriisin A_c ominaisarvot ovat

$$\lambda_{1,2} = \frac{a \pm \sqrt{a^2 + 4bc}}{2}.$$

Jaetaan ominaisarvojen tarkastelu kahteen erilliseen tapaukseen:

- $a^2 + 4bc < 0$: ominaisarvot ovat kompleksisia, ja $\mathrm{Re}(\lambda_{1,2}) = \frac{a}{2} < 0$, kun $a < 0$. Valitsemalla $a = -\alpha$ ominaisarvot ovat reaaliosaltaan aidosti negatiivisia kaikilla reaaliluvuilla $\alpha > 0$.

- $a^2 + 4bc \geq 0$: ominaisarvot ovat reaalisia. Vaaditaan ominaisarvojen "+"-vaihtoehdosta $a + \sqrt{a^2 + 4bc} < 0$, jolloin $a < -\sqrt{a^2 + 4bc} \leq 0$. Valitaan siis a aidosti negatiiviseksi: $a = -\alpha$, missä reaaliluku $\alpha > 0$. Nyt vaatimus saadaan muotoon $-\alpha + \sqrt{\alpha^2 + 4bc} < 0 \iff \alpha > \sqrt{\alpha^2 + 4bc}$, ja tämä on voimassa, kun $4bc < 0$. Valitaan siksi $b = -c$, jolloin $\sqrt{\alpha^2 - 4c^2} < \sqrt{\alpha^2} = \alpha$, ja "+"-vaihtoehdosta saadaan aidosti negatiivinen ominaisarvo. Tarkkaan ottaen, jos c olisi kompleksinen, b olisi syytä valita seuraavalla tavalla: $b = -\bar{c}$, missä $\bar{c}$ on c:n liittoluku. Tällöin $4bc = -4\bar{c}c = -4\,|c|^2 < 0$. Näillä valinnoilla myös "$-$"-vaihtoehto $-\alpha - \sqrt{\alpha^2 - 4\,|c|^2}$ on aidosti negatiivinen kahden negatiivisen luvun summana.

  Ominaisarvot ovat nyt muotoa $\lambda_{1,2} = \frac{-\alpha \pm \sqrt{\alpha^2 - 4|c|^2}}{2} < 0$. Koska parametri α on ominaisarvojen lausekkeessa kahdessa eri kohdassa, ominaisarvojen sijaintia

kompleksitasossa on jokseenkin vaikea säädellä α:n valinnoilla. Mieluummin yksinkertaistetaan lauseketta siten, että ominaisarvot riippuvat kahdesta eri parametrista, α ja k, jotka taas eivät riipu mitenkään toisistaan. Tällainen kahden parametrin lauseke saadaan valitsemalla $b = -k\alpha^2\overline{c}$, missä $k > 0$, jolloin kaikilla $\alpha > 0$ on voimassa

$$\lambda_{1,2} = \frac{-\alpha \pm \sqrt{\alpha^2 - 4k\alpha^2 |c|^2}}{2} = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k|c|^2}\right) < 0.$$

Vaikka juurrettava menisikin nyt jollain k:n valinnalla pienemmäksi kuin 0, ominaisarvojen reaaliosat olisivat kuitenkin pienempiä kuin 0.

- Yhteenveto: valinnoilla $a = -\alpha$ ja $b = -k\alpha^2\overline{c}$, missä $\alpha > 0$ ja $k > 0$, matriisin A_c ominaisarvot saadaan kompleksitasossa avoimeen vasempaan puolitasoon. Näillä valinnoilla systeemi (3.3a) on stabiili 2×2 -matriisin tapauksessa.

Seuraavalla lauseella osoitetaan, että yllä olevat valinnat yleistyvät myös yleiselle $(n+p) \times (n+p)$ -matriisille. Valitaan matriiseiksi $A = -\alpha I_n$ ja $B = -k\alpha^2\widetilde{C}^*$, missä α ja k ovat aidosti positiivisia, valittavissa olevia parametreja, ja $\widetilde{C}^*$ on myös parametrin asemassa toimiva $n \times p$ -matriisi. Perusteluna sille, miksi otettiin käyttöön matriisiparametri $\widetilde{C}^*$, on se, että näin voidaan vaikuttaa paremmin ominaisarvojen sijaintiin kompleksitasossa. Yksinkertaisin valinta parametrille lienee $\widetilde{C}^* = C^*$, mikä on analoginen 2×2 -matriisin parametrivalintojen kanssa. Kappaleessa 3.5 esitellään muita, joissain tapauksissa tehokkaampia parametrivalintoja. Tässä kappaleessa käytetään valintaa $\widetilde{C}^* = C^*$, jolloin $B = -k\alpha^2 C^*$.

Lause 3.2.2. *Olkoon matriisi A_c muotoa*

$$A_c = \begin{bmatrix} A & B \\ C & O \end{bmatrix}.$$

Jos valitaan $A = -\alpha I_n$ ja $B = -k\alpha^2 C^*$, missä $\alpha > 0$, $k > 0$ ja C^* on C:n konjugaattitranspoosi, niin matriisin A_c ominaisarvot ovat reaaliosaltaan pienempiä tai yhtä suuria kuin 0. Jos lisäksi A_c on ei-singulaarinen, niin ominaisarvojen reaaliosat ovat aidosti negatiivisia.

Todistus. Ratkaistaan ominaisarvoyhtälö

$$\begin{bmatrix} -\alpha I_n & -k\alpha^2 C^* \\ C & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}, \quad \text{missä} \quad \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \neq \mathbf{0}. \tag{3.9}$$

Saadaan yhtälöpari

$$\begin{cases} -\alpha\mathbf{u} - k\alpha^2 C^*\mathbf{v} = \lambda\mathbf{u} \\ C\mathbf{u} = \lambda\mathbf{v} \end{cases}. \tag{3.10}$$

Kerrotaan yhtälöparin toinen rivi puolittain matriisilla C^* (vasemmalta kerrottuna), jolloin toiselle riville saadaan $C^*C\mathbf{u} = \lambda C^*\mathbf{v}$, eli

$$C^*\mathbf{v} = \frac{1}{\lambda} C^*C\mathbf{u}. \tag{3.11}$$

Sijoittamalla (3.11) yhtälöparin (3.10) ensimmäiselle riville saadaan

$$\begin{aligned} & -\alpha\mathbf{u} - k\alpha^2 \frac{1}{\lambda} C^*C\mathbf{u} = \lambda\mathbf{u} \\ \iff \quad & -k\alpha^2 \frac{1}{\lambda} C^*C\mathbf{u} = (\alpha + \lambda)\mathbf{u} \\ \iff \quad & C^*C\mathbf{u} = -\frac{\lambda(\alpha+\lambda)}{k\alpha^2}\mathbf{u}. \end{aligned} \tag{3.12}$$

Huomataan, että (3.12) on matriisin C^*C ominaisarvoyhtälö, jossa $-\frac{\lambda(\alpha+\lambda)}{k\alpha^2}$ on ominaisarvo, ja $\mathbf{u}$ on ominaisarvoa vastaava ominaisvektori. Olkoon matriisilla C singulaariarvot σ_1, σ_2, …, σ_p. Koska C on $p \times n$ -matriisi ja C^* on siis kokoa $n \times p$, matriisi C^*C on kokoa $n \times n$, ja sillä on n ominaisarvoa. Näitä ominaisarvoja ovat σ_1^2, σ_2^2, …, σ_p^2 (yhteensä p kappaletta) sekä ominaisarvo 0 kertaluvulla $n - p$.

Tarkastellaan ensin matriisin C^*C ominaisarvoa σ_j^2, $j = 1, 2, \ldots, p$. Yhtälöstä (3.12) saadaan ehto

$$\begin{aligned} & -\frac{\lambda(\alpha+\lambda)}{k\alpha^2} = \sigma_j^2 \\ \iff & \lambda^2 + \alpha\lambda + k\alpha^2\sigma_j^2 = 0, \end{aligned}$$

eli

$$\lambda = \frac{-\alpha \pm \sqrt{\alpha^2 - 4k\alpha^2\sigma_j^2}}{2} = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_j^2}\right). \tag{3.13}$$

Juurrettavalle on voimassa $1 - 4k\sigma_j^2 < 1$, ja tästä seuraa, että $\mathrm{Re}(\lambda) < 0$.

Tarkastellaan seuraavaksi matriisin C^*C ominaisarvoa 0. Yhtälöstä (3.12) saadaan ehto

$$\begin{aligned} & -\frac{\lambda(\alpha+\lambda)}{k\alpha^2} = 0 \\ \iff \quad & \lambda^2 + \alpha\lambda = 0. \end{aligned}$$

Siis $\lambda = 0$ tai $\lambda = -\alpha < 0$. Jos matriisi A_c on ei-singulaarinen, niin ratkaisuksi kelpuutetaan vain ominaisarvo $\lambda = -\alpha$. Tällöin kaikki A_c:n ominaisarvot ovat

reaaliosaltaan aidosti negatiivisia. □

Lauseesta 3.2.2 huomataan, että valitsemalla $A = -\alpha I_n$ ja $B = -k\alpha^2 C^*$ systeemi (3.3a) on aina stabiili, kunhan varmistutaan siitä, että matriisi A_c on ei-singulaarinen.

Lause 3.2.3. *Olkoon A_c lohkomatriisi*

$$A_c = \begin{bmatrix} A & B \\ C & O \end{bmatrix}, \tag{3.14}$$

missä $A = -\alpha I_n$ on $n \times n$ -diagonaalimatriisi, $B = -k\alpha^2 C^*$ on $n \times p$ -matriisi ($p \leq n$) ja C on $p \times n$ -matriisi, ja lisäksi $\alpha > 0$ ja $k > 0$. Matriisi A_c on ei-singulaarinen, jos ja vain jos $\operatorname{rank}(C)$ *on täysi.*

Todistus. Matriisi $A = -\alpha I_n$ on selvästi ei-singulaarinen. Tällöin viitteen [20] mukaan matriisi A_c voidaan esittää kahden lohkokolmiomatriisin tulona:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & O \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ O & D - CA^{-1}B \end{bmatrix}. \tag{3.15}$$

Matriisin A_c determinantti on tällöin

$$\det(A_c) = \det\left(\begin{bmatrix} I & O \\ CA^{-1} & I \end{bmatrix}\right) \det\left(\begin{bmatrix} A & B \\ O & D - CA^{-1}B \end{bmatrix}\right). \tag{3.16}$$

Koska matriisi $\begin{bmatrix} I & O \\ CA^{-1} & I \end{bmatrix}$ on alakolmiomatriisi, jonka diagonaalialkiot ovat ykkösiä, sen determinantti on 1. Siis yhtälössä (3.16) oikean puolen jälkimmäinen matriisi määrää A_c:n determinantin. Koska $\begin{bmatrix} A & B \\ O & D - CA^{-1}B \end{bmatrix}$ on lohkoyläkolmiomatriisi, sen determinantti on viitteiden [10] ja [20] nojalla diagonaalilohkojen determinanttien tulo:

$$\begin{aligned} \det(A_c) &= \det\left(\begin{bmatrix} A & B \\ O & D - CA^{-1}B \end{bmatrix}\right) \\ &= \det(A)\det(D - CA^{-1}B) \\ &\overset{D=O}{=} \det(A)\det(-CA^{-1}B) \\ &= (-1)^p \det(A)\det(CA^{-1}B). \end{aligned} \tag{3.17}$$

Koska $\det(A) \neq 0$, niin $\det(A_c) \neq 0$ täsmälleen silloin, kun $\det(CA^{-1}B) \neq 0$. Matriisilla $CA^{-1}B$ on siis oltava täysi aste. Koska $B = -k\alpha^2 C^*$, matriisin $CA^{-1}B$

aste on täysi täsmälleen silloin, kun C:n aste on täysi. Toisin sanottuna, matriisi A_c on kääntyvä, jos ja vain jos $\text{rank}(C) = p$. □

Jatkossa tämä aste-ehto voidaan olettaa todeksi, koska lineaarisen yhtälöryhmän ratkaisun kannalta ollaan kiinnostuneita vain niistä tilanteista, joissa yhtälöryhmällä on olemassa ratkaisu. Silloinhan ratkaisua ei yleensä ole, kun $\text{rank}(C) < p$.

Valitsemalla systeemin (3.3a) matriisit A ja B lauseen 3.2.2 mukaisesti matriisin

$$A_c = \begin{bmatrix} -\alpha I_n & -k\alpha^2 C^* \\ C & O \end{bmatrix} \tag{3.18}$$

ominaisarvot ovat $\frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_j^2}\right)$, $j = 1, 2, \ldots, p$ (yhteensä $2p$ kappaletta), sekä $-\alpha$ kertaluvulla $n - p$. Matriisilla A_c on siis yhteensä $2p + n - p = n + p$ kappaletta ominaisarvoja. Lausekkeesta (3.13) seuraa, että osa ominaisarvoista voi olla kompleksisia. Näin käy, kun

$$1 - 4k\sigma_j^2 < 0 \iff k > \frac{1}{4\sigma_j^2} \, . \tag{3.19}$$

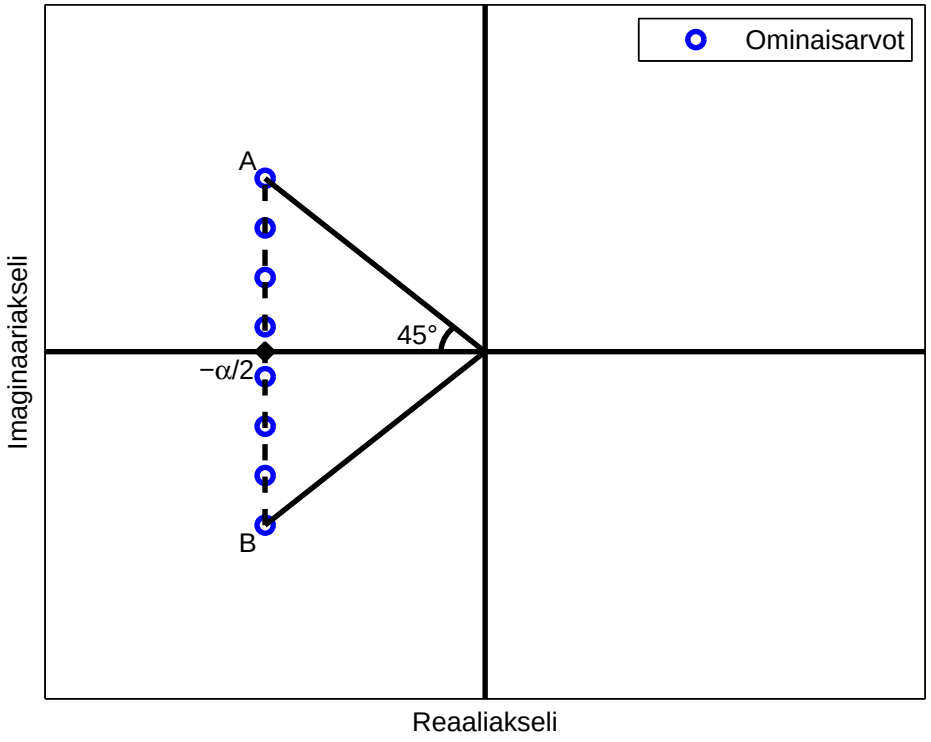
Kompleksiset ominaisarvot aiheuttavat värähtelyä vektorin $\mathbf{x}_c(t)$ ratkaisuun. Mitä suuremmat ominaisarvojen imaginaariosat ovat, sitä voimakkaampaa värähtely on.

Suuri värähtely vektorissa $\mathbf{x}_c(t)$ hidastaa differentiaaliyhtälöiden ratkaisijoiden toimintaa. Sen takia ominaisarvot pyritään saamaan kompleksitasossa esimerkiksi $\pm 45°$ sektoriin reaaliakselista katsottuna kuvan 3.2 mukaisella tavalla. Osalle ominaisarvoista on voimassa yhtälön (3.13) mukainen lauseke

$$\lambda = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_j^2}\right), \quad j = 1, 2, \ldots, p$$

ja loput $n - p$ ominaisarvoa saavat arvon $-\alpha$. Tapauksessa $\lambda = -\alpha$ ominaisarvot kuuluvat $\pm 45°$ sektoriin reaaliakselista katsottuna — itse asiassa reaaliset ominaisarvot kuuluvat aina $\pm 45°$ sektoriin, koska ne ovat kulmassa $0°$ reaaliakseliin nähden. Jos yhtälössä (3.13) diskriminantille on voimassa $1 - 4k\sigma_j^2 \geq 0$, myös yhtälön (3.13) mukaiset ominaisarvot ovat reaalisia ja kuuluvat $\pm 45°$ sektoriin. Tarkastellaan vielä tapaus $1 - 4k\sigma_j^2 < 0$. Jotta ominaisarvot pysyisivät kuvan 3.2 mukaisella janalla AB, on oltava

$$\begin{aligned} -\frac{\alpha}{2}\mathrm{i} \; &< \frac{\alpha}{2}\sqrt{1 - 4k\sigma_j^2} \; &< \frac{\alpha}{2}\mathrm{i} \\ \iff -\frac{\alpha}{2}\mathrm{i} \; &< \frac{\alpha}{2}\mathrm{i}\sqrt{4k\sigma_j^2 - 1} \; &< \frac{\alpha}{2}\mathrm{i} \\ \iff \; -1 \; &< \; \sqrt{4k\sigma_j^2 - 1} \; &< 1 \; , \end{aligned}$$



Kuva 3.2: Ominaisarvot halutaan pitää ±45° sektorissa reaaliakseliin nähden.

ja korottamalla puolittain toiseen saadaan

$$\begin{aligned} & 4k\sigma_j^2 - 1 < 1 \\ \iff \qquad & k < \frac{1}{2\sigma_j^2} \ . \end{aligned} \tag{3.20}$$

Jos matriisin A_c ominaisarvot pysyvät kuvan 3.2 mukaisella janalla AB (ja loput $n - p$ ominaisarvoa saavat arvon $-\alpha$), värähtely jää yleensä suhteellisen pieneksi. Lisäksi ominaisarvojen reaaliosat ovat hyvin lähellä toisiaan: $(2p)$:llä ominaisarvolla reaaliosa on $-\frac{\alpha}{2}$, ja $(n-p)$:llä ominaisarvolla reaaliosa on $-\alpha$. Määritelmän 2.2.9 mukaan systeemi ei siis myöskään ole jäykkä.

Jotta $2p$ kappaletta ominaisarvoja olisi suoralla AB, ehdon (3.19) on toteuduttava kaikilla j:n arvoilla, $j = 1, 2, \ldots, p$. Siis on oltava

$$k > \frac{1}{4\sigma_p^2}.$$

Tällöin $2p$ kappaletta ominaisarvoista on kompleksisia. Tässä on oletettu, että matriisin C singulaariarvot $\sigma_1, \sigma_2, \ldots, \sigma_p$ on indeksoitu suuruusjärjestyksesä suurimmasta pienimpään. Jotta nämä $2p$ ominaisarvoa pysyisivät myös ±45° sektorissa

reaaliakseliin nähden, ehdon (3.20) on oltava voimassa kaikilla j:n arvoilla. On siis oltava

$$k < \frac{1}{2\sigma_1^2}.$$

Kun vaatimukset kompleksisuudesta ja $\pm 45°$ sektoriin kuulumisesta yhdistetään, parametrin k täytyy kuulua välille

$$\frac{1}{4\sigma_p^2} < k < \frac{1}{2\sigma_1^2}. \tag{3.21}$$

Jotta ehto (3.21) toteutuisi jollain k:n arvolla, on oltava voimassa

$$\frac{1}{4\sigma_p^2} < \frac{1}{2\sigma_1^2} \quad \Longleftrightarrow \quad 4\sigma_p^2 > 2\sigma_1^2 \quad \Longleftrightarrow \quad \frac{\sigma_1^2}{\sigma_p^2} < 2,$$

eli

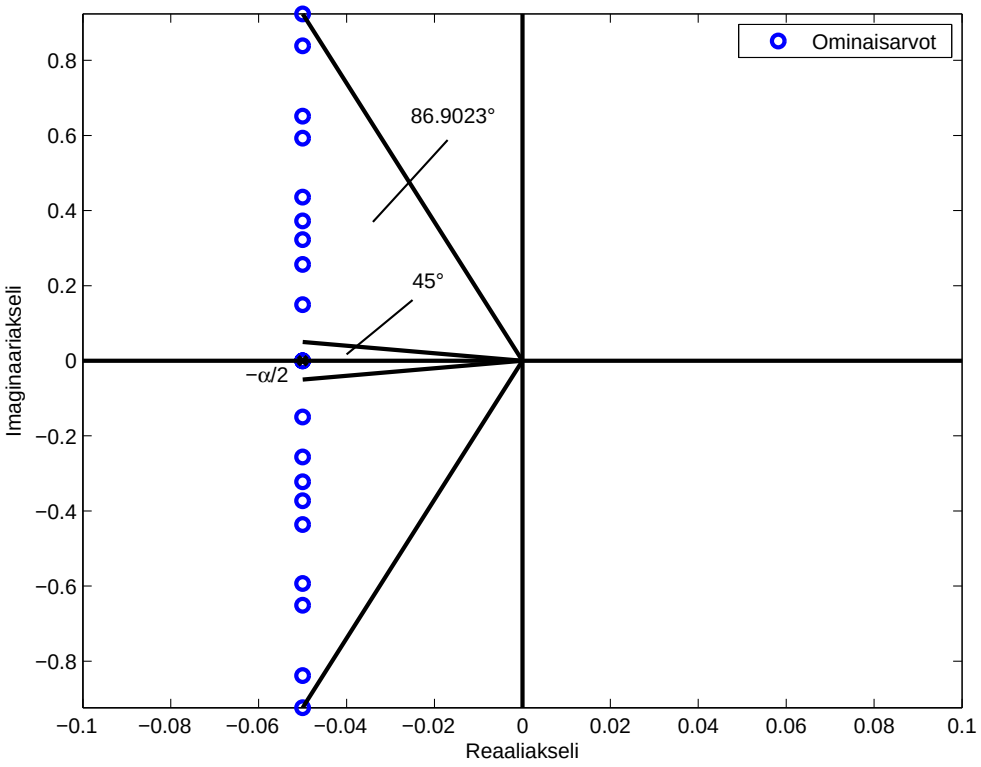
$$\mathrm{cond}(C) = \frac{\sigma_1}{\sigma_p} < \sqrt{2}. \tag{3.22}$$

Siis matriisin C kuntoluvun on oltava pienempi kuin $\sqrt{2}$. Käytännössä tämä ehto toteutuu hyvin harvoin. Toisin sanottuna, on hyvin vaikea päästä sellaiseen tilanteeseen, jossa $2p$ ominaisarvoa ovat sekä kompleksisia että $\pm 45°$ sektorissa. Jos parametria k kasvatetaan, ominaisarvojen kompleksisuus toteutuu, mutta samalla ominaisarvot "karkaavat" pois $\pm 45°$ sektorista (kuva 3.3), jolloin värähtely alkaa aiheuttaa ongelmia laskennassa. Jos taas parametria k pienennetään, niin $\pm 45°$ sektorissa pysytään, mutta ominaisarvot alkavat levitä negatiiviselle reaaliakselille. Tällöin ominaisarvojen reaaliosat alkavat poiketa toisistaan merkittävästi (kuva 3.4), ja systeemistä tulee jäykkä, mikä ei myöskään ole hyvä tilanne laskennan kannalta. Parametria k asetettaessa on usein valittava joko jäykkyys tai värähtely.

Tässä vaiheessa differentiaaliyhtälösysteemi (3.3) on teoriassa valmis ratkaistavaksi. Lauseen 2.2.4 nojalla tiedetään, että systeemin $\mathbf{x}_c'(t) = A_c\mathbf{x}_c(t) + \mathbf{r}_c$ ratkaisu alkuehdolla $\mathbf{x}_c(0) = \mathbf{x}_{c,0}$ on

$$\mathbf{x}_c(t) = \mathrm{e}^{tA_c}(\mathbf{x}_{c,0} + A_c^{-1}\mathbf{r}_c) - A_c^{-1}\mathbf{r}_c.$$

Ratkaisua ei kuitenkaan kannata laskea matriisieksponentin avulla vaan käyttämällä numeerisia ratkaisijoita, kuten `ode45` tai `ode15s`. Näille funktioille voidaan asettaa parametrina lopetusehto siten, että hyväksytään ratkaisuksi ensimmäinen havaittu $\mathbf{x}(t)$, jolle ehto (3.6) toteutuu. Seuraavissa luvuissa DY-ratkaisijan parametreja $\widetilde{C}^*$, k ja α pyritään optimoimaan, jotta differentiaaliyhtälö saataisiin ratkaistua mahdollisimman nopeasti.



Kuva 3.3: Matriisin A_c ominaisarvot kompleksitasossa satunnaisen 10×10 -matriisin C tapauksessa, kun $\alpha = 0{,}1$ ja $k = \frac{1}{4\sigma_p^2} = 3{,}41$ ovat kiinteitä. Ominaisarvot "karkaavat" pois $\pm 45°$ sektorista aiheuttaen värähtelyä.

3.3 Matriisin leveyden merkitys

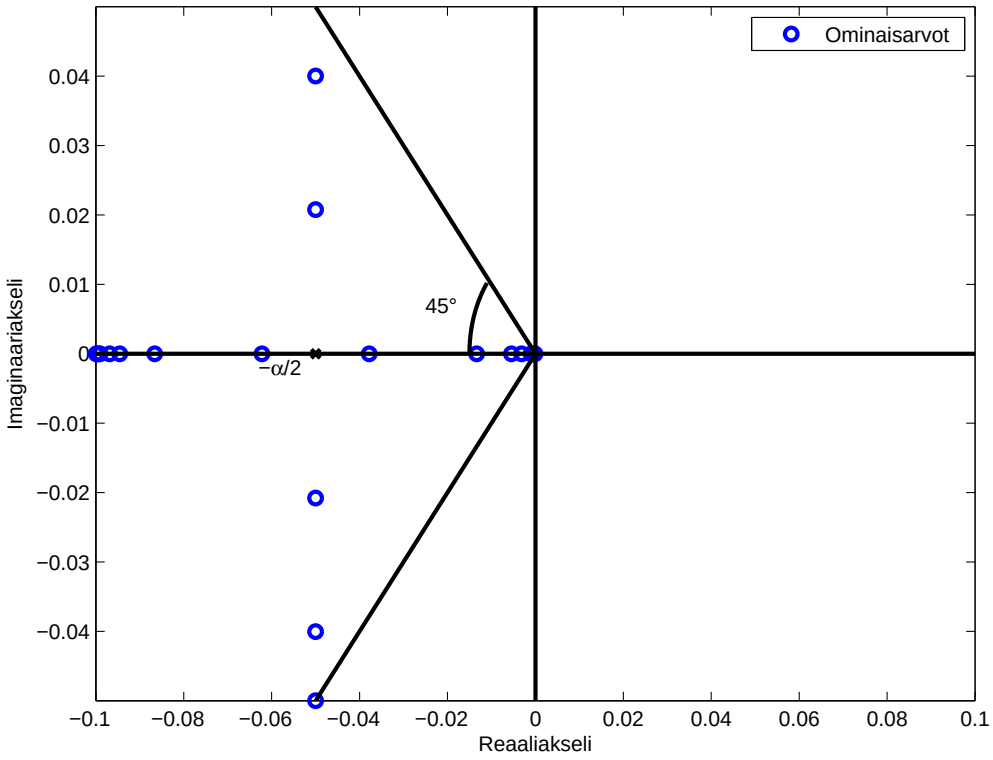
Tässä luvussa johdetaan yllättävä tulos, jonka mukaan leveällä matriisilla C systeemin (3.3) ratkaiseminen on laskennallisesti helpompaa kuin neliömatriisilla C. Seuraavan lauseen mukaan leveillä matriiseilla on tavallisesti suuremmat singulaariarvot kuin neliömatriiseilla.

Lause 3.3.1. *Olkoon C_1 $p \times p$ -neliömatriisi ja C leveä $p \times n$ -matriisi, joka on ositettu seuraavalla tavalla:*

$$C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}.$$

Jos C_1:n singulaariarvot ovat α_1, α_2, …, α_p, niin C:n singulaariarvot ovat muotoa $\sigma_i = \alpha_i + M_i$, missä $M_i \geq 0$ ja $i = 1, 2, \ldots, p$. Toisin sanottuna, matriisia levennettäessä sen singulaariarvot kasvavat.

Todistus. Olkoon C leveä $p \times n$ -matriisi. Muodostetaan sille ositus $C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$, missä C_1 on $p \times p$ -neliömatriisi ja C_2 on kokoa $p \times (n - p)$. Selvästi matriisi CC^* on hermiittinen, ja sen ominaisarvoja ovat C:n singulaariarvojen neliöt, σ_i^2, $i = 1$,



Kuva 3.4: Matriisin A_c ominaisarvot kompleksitasossa satunnaisen 10×10 -matriisin C tapauksessa, kun $\alpha = 0{,}1$ ja $k = \frac{1}{2\sigma_1^2} = 0{,}0176$ ovat kiinteitä. Kaikki ominaisarvot kuuluvat $\pm 45°$ sektoriin, mutta osa reaalisista ominaisarvoista on hyvin lähellä imaginaariakselia aiheuttaen jäykkyyttä systeemiin. Nyt $\min_i |\mathrm{Re}(\lambda_i)| = 9{,}53 \cdot 10^{-5}$ ja $\max_i |\mathrm{Re}(\lambda_i)| = 9{,}99 \cdot 10^{-2}$, jolloin määritelmässä 2.2.9 mainittu suhde on $1{,}05 \cdot 10^3$. Systeemiä voidaan siis pitää melko jäykkänä. Jos parametria k kasvatetaan, reaaliset ominaisarvot siirtyvät reaaliakselia pitkin kohti arvoa $-\frac{\alpha}{2}$, ja kasvattamalla k:ta edelleen ominaisarvot alkavat kuvan 3.3 tavoin kulkea pitkin suoraa $x = -\frac{\alpha}{2} \pm \mathrm{i}b$, missä $b \to \infty$, kun $k \to \infty$.

2, ..., p. Muodostetaan ominaisarvoyhtälö $CC^*\mathbf{x} = \lambda\mathbf{x}$:

$$\begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} C_1^* \\ C_2^* \end{bmatrix} \mathbf{x} = \sigma_i^2 \mathbf{x},$$

tai aukikirjoitettuna

$$(C_1 C_1^* + C_2 C_2^*)\mathbf{x} = \sigma_i^2 \mathbf{x}. \tag{3.23}$$

Merkitään $\mathcal{A} = C_1 C_1^*$, $\mathcal{B} = C_2 C_2^*$ ja $\mathcal{C} = C_1 C_1^* + C_2 C_2^* = \mathcal{A} + \mathcal{B}$, jolloin (3.23) on $\mathcal{C}$:n ominaisarvoyhtälö

$$\mathcal{C}\mathbf{x} = \sigma_i^2 \mathbf{x},$$

ja σ_i^2 on $\mathcal{C}$:n ominaisarvo. Selvästi matriisit $\mathcal{A} = C_1 C_1^*$, $\mathcal{B} = C_2 C_2^*$ ja $\mathcal{C} = CC^*$ ovat

hermiittisiä, ja on helppo näyttää, että $\mathcal{A}$, $\mathcal{B}$ ja $\mathcal{C}$ ovat positiivisesti semidefiniittejä:

$$\langle \mathbf{x},\, \mathcal{A}\mathbf{x}\rangle = \langle \mathbf{x},\, C_1C_1^*\mathbf{x}\rangle \overset{(2.10)}{=} \langle C_1^*\mathbf{x},\, C_1^*\mathbf{x}\rangle = \|C_1^*\mathbf{x}\|^2 \geq 0 \quad \forall \mathbf{x}.$$

Matriisien $\mathcal{A}$, $\mathcal{B}$ ja $\mathcal{C}$ ominaisarvot ovat siis suurempia tai yhtä suuria kuin 0. Merkitään $\mathcal{A}$:n ja $\mathcal{B}$:n ominaisarvoja α_i^2 ja β_i^2, vastaavasti, missä $i = 1, 2, \ldots, p$. Koska $\mathcal{A}$, $\mathcal{B}$ ja $\mathcal{C}$ ovat hermiittisiä, viitteen [31, s. 101–102] nojalla $\mathcal{C}$:n ominaisarvoille σ_i^2 on voimassa:

$$\alpha_i^2 + \min_j \beta_j^2 \leq \sigma_i^2 \leq \alpha_i^2 + \max_j \beta_j^2.$$

Matriisin $\mathcal{B}$ positiivisesta semidefiniittisyydestä seuraa erityisesti

$$\sigma_i^2 \geq \alpha_i^2 + \min_j \beta_j^2 \geq \alpha_i^2, \tag{3.24}$$

missä i ja j saavat arvot 1, 2, …, p.

Matriisilla $\mathcal{C} = CC^*$ on siis suuremmat tai yhtä suuret ominaisarvot kuin matriisilla $\mathcal{A} = C_1C_1^*$. Huomaamalla, että $\mathcal{A}$:n ja $\mathcal{C}$:n ominaisarvot ovat vastaavasti C_1:n ja C:n singulaariarvoja, joita merkitään vastaavasti α_i ja σ_i, saadaan tulos, että leveällä matriisilla $\begin{bmatrix} C_1 & C_2 \end{bmatrix}$ on aina suuremmat tai yhtä suuret singulaariarvot kuin pelkällä neliömatriisilla C_1. Juuri tämän takia leveillä matriiseilla on suuremmat singulaariarvot kuin neliömatriiseilla. Jos C_2:n singulaariarvot ovat suuruusjärjestyksessä $\beta_1 \geq \beta_2 \geq \ldots \geq \beta_p$, niin aiempia merkintöjä käyttäen C:n singulaariarvot ovat muotoa $\sigma_i = \alpha_i + M_i$, missä $M_i \in [\beta_p, \beta_i]$. Näin saatiin myös teoreettinen vaihteluväli lisäykselle M_i. □

Tarkastellaan seuraavaksi matriisin A_c ominaisarvoja. Ne ovat yhtälön (3.13) perusteella muotoa

$$\lambda = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_j^2}\right), \quad j = 1, 2, \ldots, p,$$

kun valitaan $\widetilde{C}^* = C^*$. Parametri α on vain kiinteä skaalauskerroin, joten parametri k sekä C:n singulaariarvot σ_j määräävät ominaisarvojen sijainnin kompleksitasossa. Luvussa 3.4.1 johdetaan tulos, että tehokkain tapa valita k `ode45`-ratkaisijalle on

$$k = \frac{1}{8}\left(\frac{1}{\sigma_{p-1}^2} + \frac{1}{\sigma_p^2}\right).$$

Kun k on valittu tällä tavalla pienimpien singulaariarvojen avulla, matriisin A_c ominaisarvoista vain juuripari $\frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_p^2}\right)$ on reaalinen. Tällöin juurrettavalle

on voimassa

$$\begin{aligned} 1 - 4k\sigma_p^2 &= 1 - \frac{4}{8}\left(\frac{\sigma_p^2}{\sigma_{p-1}^2} + \frac{\sigma_p^2}{\sigma_p^2}\right) \\ &= 1 - \frac{1}{2}\left(1 + \frac{\sigma_p^2}{\sigma_{p-1}^2}\right) \\ &= \frac{1}{2}\left(1 - \frac{\sigma_p^2}{\sigma_{p-1}^2}\right). \end{aligned}$$

Koska $\frac{\sigma_p^2}{\sigma_{p-1}^2} \leq 1$, on juurrettava suurempi tai yhtä suuri kuin 0. Jos neliömatriisilla on singulaariarvot σ_1, σ_2, …, σ_p, niin leveällä matriisilla singulaariarvot ovat Lauseen 3.3.1 nojalla muotoa $\sigma_1 + M_1$, $\sigma_2 + M_2$, …, $\sigma_p + M_p$, missä $M_i \geq 0$. Mitä leveämpi matriisi, sitä suurempia termit M_i ovat. Kun matriisin leveys $n \gg p$ on hyvin suuri, myös M_i on suur, jolloin suhde $\frac{(\sigma_p+M_p)^2}{(\sigma_{p-1}+M_{p-1})^2}$ on lähellä arvoa 1:

$$\frac{\sigma_p^2}{\sigma_{p-1}^2} \leq \frac{(\sigma_p + M_p)^2}{(\sigma_{p-1} + M_{p-1})^2} \overset{M_i \text{ suuri}}{\approx} \frac{M_p^2}{M_{p-1}^2} \approx 1.$$

Tällöin juurrettava $\frac{1}{2}\left(1 - \frac{\sigma_p^2}{\sigma_{p-1}^2}\right)$ lähestyy nollaa, ja matriisin A_c ominaisarvo λ lähestyy arvoa $-\frac{\alpha}{2}$. Mitä suurempia singulaariarvot σ_{p-1} ja σ_p siis ovat, sitä pienempi juurrettava on, ja sitä lähempänä λ on arvoa $-\frac{\alpha}{2}$. Siksi suurten singulaariarvojen tapauksessa reaaliset ominaisarvot eivät ole lähellä imaginaariakselia, eivätkä systeemit ole yhtä jäykkiä leveillä matriiseilla C kuin neliömatriiseilla C.

Tarkastellaan seuraavaksi matriisin A_c kompleksisia ominaisarvoja. Valitsemalla k edelleen muodossa $k = \frac{1}{8}\left(\frac{1}{\sigma_{p-1}^2} + \frac{1}{\sigma_p^2}\right)$ matriisin A_c ominaisarvoilla on $p-1$ kompleksista juuriparia:

$$\lambda = \frac{\alpha}{2}\left(-1 \pm \mathrm{i}\sqrt{4k\sigma_j^2 - 1}\right), \quad j = 1, 2, \ldots, p-1.$$

Nyt riittää tarkastella imaginaariosan lauseketta $4k\sigma_j^2 - 1$. Sijoittamalla k:n lauseke päästään muotoon

$$\frac{4}{8}\left(\frac{\sigma_j^2}{\sigma_{p-1}^2} + \frac{\sigma_j^2}{\sigma_p^2}\right) - 1 = \frac{1}{2}\left(\frac{\sigma_j^2}{\sigma_{p-1}^2} + \frac{\sigma_j^2}{\sigma_p^2} - 2\right).$$

Huomataan, että $\frac{\sigma_j^2}{\sigma_{p-1}^2} \geq 1$ ja $\frac{\sigma_j^2}{\sigma_p^2} \geq 1$, kun j saa arvot 1, 2, …, $p-1$. Jälleen Lauseen 3.3.1 nojalla leveällä matriisilla singulaariarvot ovat lisäyksen $M_i \geq 0$ verran suuremmat kuin neliömatriisilla, $i = 1, 2, \ldots, p$. Kun matriisia levennetään ja $n \gg p$

lähestyy ääretöntä, myös M_i lähestyy ääretöntä, jolloin suhteet

$$\frac{(\sigma_j + M_j)^2}{(\sigma_{p-1} + M_{p-1})^2} \quad \text{ja} \quad \frac{(\sigma_j + M_j)^2}{(\sigma_p + M_p)^2}$$

lähestyvät arvoa 1 kaikilla j:n arvoilla 1, 2, ..., $p-1$. Siis

$$\frac{\sigma_j^2}{\sigma_{p-1}^2} \geq \frac{(\sigma_j + M_j)^2}{(\sigma_{p-1} + M_{p-1})^2} \overset{M_i \text{ suuri}}{\approx} 1 \qquad \text{ja} \qquad \frac{\sigma_j^2}{\sigma_p^2} \geq \frac{(\sigma_j + M_j)^2}{(\sigma_p + M_p)^2} \overset{M_i \text{ suuri}}{\approx} 1,$$

jolloin edelleen

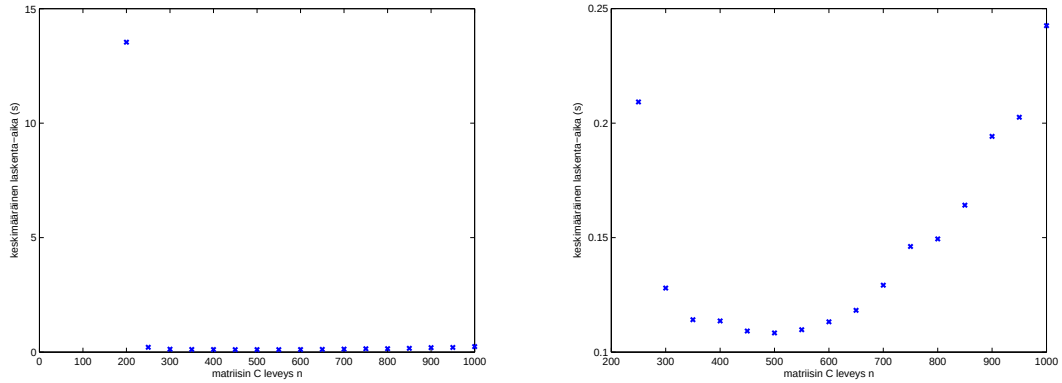
$$\frac{1}{2}\left(\frac{(\sigma_j + M_j)^2}{(\sigma_{p-1} + M_{p-1})^2} + \frac{(\sigma_j + M_j)^2}{(\sigma_p + M_p)^2} - 2\right) \approx \frac{1}{2}(1 + 1 - 2) = 0,$$

ja ominaisarvon koko imaginaariosa lähestyy nollaa. Mitä suurempia singulaariarvot σ_j siis ovat, $j = 1, 2, \ldots, p$ — huomaa, että nyt tarvitaan myös indeksi p — sitä lähempänä nollaa A_c:n ominaisarvojen imaginaariosat ovat. Tästä johtuen suurten singulaariarvojen tapauksessa kompleksisilla ominaisarvoilla on itseisarvoltaan pienet imaginaariosat, ja siksi systeemissä esiintyy vähemmän värähtelyä leveillä matriiseilla C kuin neliömatriiseilla C. Huomaa, että samat tulokset pätevät millä tahansa k:n valinnalla: reaaliset ominaisarvot eivät mene lähelle imaginaariakselia, ja kompleksisten ominaisarvojen imaginaariosat ovat pienet.

3.3.1 Simulaatioita

Havainnollistetaan simulaatioiden avulla, miten merkittävästi differentiaaliyhtälön ratkaisu helpottuu, jos C on leveä matriisi. Olkoon C kokoa $p \times n$, missä $p \leq n$. Annetaan matriisin korkeuden p olla kiinteä ja katsotaan, miten systeemin (3.3) ratkaisuun käytettävä laskenta-aika muuttuu, kun C:n leveyttä n kasvatetaan. Simulaatioissa asetetaan $p = 200$, ja n saa arvot $n_1 = 200$ $(= p)$, $n_2 = 250$, ..., $n_{17} = 1000$. Käytetään ensin `ode45`-ratkaisijaa ja valitaan parametrit α ja k kokeilemalla siten, että ratkaisijan laskenta-aika on mahdollisimman lyhyt. Lisäksi valitaan $\widetilde{C}^* = C^*$. Laskenta-aika kullekin matriisikoolle saadaan siten, että generoidaan 1000 kappaletta $p \times n_i$ -kokoisia matriiseja ja lasketaan systeemin ratkaisuun kuluva aika laskenta-aikojen keskiarvona, $i = 1, 2, \ldots, 17$. Ajan säästämiseksi asetetaan jälleen `ode45`-ratkaisijalle aikarajaksi 15 sekuntia. Simulaatioiden tulokset on esitetty kuvassa 3.5.

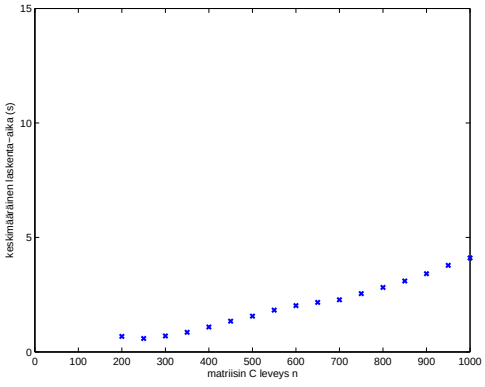
Huomataan, että neliömatriiseilla laskenta on huomattavasti hitaampaa kuin leveillä matriiseilla. Jos tarkastellaan matriisin A_c ominaisarvojen käyttäytymistä, havaitaan kaksi ominaisuutta. Ensinnäkin, C:n ollessa leveä matriisi A_c:n reaaliset ominaisarvot eivät siirry läheskään yhtä lähelle imaginaariakselia kuin C:n olles-



Kuva 3.5: Ratkaisijan ode45 laskenta-ajat $p \times n$ -matriisille C, missä matriisin korkeus $p = 200$ ja leveyttä n kasvatetaan. Kullakin leveydellä n_i laskenta-aika on saatu 1000 simulaation keskiarvona, $n_i = 200, 250, \ldots, 1000$. Koska 200×200 -neliömatriisilla laskenta-aika on keskimäärin selvästi korkeampi kuin leveillä matriiseilla, oikeaan kuvaajaan on piirretty laskenta-ajat ainoastaan leveille matriiseille (huomaa, että oikeassa kuvaajassa y-akselin asteikko on erilainen kuin vasemmassa!). Keskimääräinen laskenta-aika vaikuttaisi lyhimmältä, kun $2p \leq n \leq 3p$.

sa neliömatriisi. Tällöin systeemin jäykkyys vähenee. Toiseksi, leveiden matriisien tapauksessa kompleksisten ominaisarvojen imaginaariosat eivät kasva yhtä suuriksi kuin neliömatriiseilla, jolloin myös aaltoilu vähenee.

Tarkastellaan seuraavaksi, miten `ode15s`-ratkaisijan keskimääräiset laskenta-ajat muuttuvat, kun kokoa $p \times n$ olevan matriisin C leveyttä n kasvatetaan. Valitaan $\widetilde{C}^* = C^*$, ja parametrit α ja k valitaan kokeilemalla. Ajan säästämiseksi asetetaan `ode15s`:lle aikarajaksi 15 sekuntia. Laskenta-ajat matriisien eri leveyksillä n_i on esitetty kuvassa 3.6. Kuten `ode45`:n simulaatioissa, kunkin leveyden laskenta-aika on saatu 1000 simulaation keskiarvona.



Kuva 3.6: Ratkaisijan ode15s laskenta-ajat $p \times n$ -matriisille C, missä matriisin korkeus $p = 200$ ja leveyttä n kasvatetaan. Kullakin leveydellä n_i laskenta-aika on saatu 1000 simulaation keskiarvona, $n_i = 200, 250, \ldots, 1000$. Toisin kuin ode45:tä käytettäessä, keskimääräinen laskenta-aika kasvaa, kun leveyttä n kasvatetaan.

Huomataan, että `ode15s` ratkaisee differentiaaliyhtälön huomattavasti nopeam-

min kuin `ode45`, kun C on neliömatriisi. Tämä johtuu siitä, että neliömatriisien tapauksessa systeemi on jäykkä. Ratkaisijoista `ode15s` on nimenomaan suunniteltu jäykille systeemeille, ja kun parametrit α ja k on vielä valittu siten, että ratkaisija toimisi mahdollisimman hyvin, `ode15s`:stä saadaan tehokas ratkaisija niille systeemeille, joissa C on neliömatriisi. Laskenta-aika kasvaa vähitellen, kun C:n leveyttä n kasvatetaan.

Koska `ode45` on huomattavasti tehokkaampi leveillä matriiseilla kuin neliömatriiseilla, herää kysymys, kuinka paljon suurempi matriisin leveyden n on oltava korkeuteen p verrattuna, jotta `ode45` toimisi tehokkaasti. Ajetaan ylimääräisiä simulaatioita $200 \times (200+i)$ -matriiseille, missä $i = 1, 2, \ldots, 20$. Huomataan, että ratkaisija on jo selvästi nopeampi kuin neliömatriisin tapauksessa, kun $i = 2$ tai $i = 3$. Kun $i \geq 10$, `ode45`-ratkaisijan laskenta-ajat alkavat olla samaa luokkaa kuin `ode15s`:n, ja kun $i \geq 15$, `ode45` on jo selvästi nopeampi kuin `ode15s`.

Simulaatioiden perusteella `ode15s`-ratkaisija on tehokkaampi kuin `ode45`, kun C on neliömatriisi. Muussa tapauksessa `ode45` on tehokkaampi. Käytettävä ratkaisija voidaan valita esimerkiksi seuraavan periaatteen mukaan, kun C on kokoa $p \times n$:

- `ode15s`, kun $0 \leq n - p < 10$.
- `ode45`, kun $n - p \geq 10$.

3.4 Parametrien valinta

Tässä kappaleessa on käsitelty, miten parametrit $k > 0$ ja $\alpha > 0$ kannattaa valita, jotta differentiaaliyhtälöiden ratkaisufunktiot `ode45` sekä `ode15s` toimisivat mahdollisimman tehokkaasti. Parametri $\widetilde{C}^*$ on valittu tässä luvussa seuraavalla tavalla: $\widetilde{C}^* = C^*$. Yhtälön (3.13) perusteella α on vain skaalauskerroin A_c:n ominaisarvoille. Oletetaan siksi, että α:lle voidaan valita jokin kiinteä arvo, joka ei riipu parametreista k ja $\widetilde{C}^*$. Sen sijaan juurrettavassa esiintyvä lauseke $(4k\sigma_j^2 - 1)$ riippuu kummastakin parametrista k ja $\widetilde{C}^*$. On tärkeää selvittää k:lle sopiva arvo, jotta matriisin A_c ominaisarvot saadaan asetettua haluttuun paikkaan kompleksitasossa, ja näin systeemin jäykkyyttä ja värähtelyä voidaan säädellä. Luvussa 3.5 käsitellään vaihtoehtoisia tapoja valita $\widetilde{C}^*$, ja niillä valinnoilla myös parametri k on muodostettava eri tavalla.

3.4.1 Parametri k

Tarkastellaan differentiaaliyhtälösysteemiä (3.3) tapauksessa, jossa A_c on valittu kaavan (3.18) mukaisesti. Nyt siis parametri $\widetilde{C}^*$ on valittu C:n konjugaattitranspoosiksi. Jos systeemissä on paljon värähtelyä, esimerkiksi jos ominaisarvot ovat imaginaarisia ($k > \frac{1}{4\sigma_p^2}$) mutta eivät kuulu $\pm 45°$ sektoriin, niin differentiaaliyhtälön

ratkaiseminen on hyvin hidasta DY-ratkaisijoilla. Tarkastellaan tapauksia, joissa C on normaalijakaumaa N(0, 1) noudattava satunnaisesti generoitu 100×100 -matriisi ja joissa parametri k on valittu seuraavalla tavalla:

$$k = \frac{1}{4\sigma_p^2},$$

missä σ_p on C:n pienin singulaariarvo. Valitaan parametrille α kokeellinen arvo 0,1. Merkitään T:llä aikaa, joka DY-ratkaisijalla kuluu yhtälön $C\mathbf{x} = \mathbf{r}$ ratkaisemiseen toleranssilla $\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}$. Matriisin C kuntoluku cond(C), parametri k ja laskenta-aika T MATLAB:in ratkaisijafunktioille `ode45` ja `ode15s` on esitetty taulukossa 3.1.

Taulukko 3.1: Laskenta-aikoja C:n ollessa satunnaisesti generoitu 100×100 -matriisi ja systeemin ollessa voimakkaasti värähtelevä. Sarakkeissa on lueteltuna C:n kuntoluku, parametrin k arvo, ode45-ratkaisijan laskenta-aika sekunteina ja ode15s-ratkaisijan laskenta-aika sekunteina 10 eri matriisille. Värähtelevälle systeemille ode45 on nopeampi ratkaisija.

cond(C)	k	T_{ode45} (s)	T_{ode15s} (s)
140,2	11,9	6,1	24,7
149,6	15,9	9,8	22,5
194,3	24,0	8,5	36,0
244,6	39,7	11,4	36,8
399,8	102,2	25,1	76,2
471,4	151,4	56,2	248,0
693,3	312,8	77,4	162,3
805,4	431,1	108,0	165,2
1218,5	947,3	237,4	504,0
1348,6	1205,9	255,1	333,2
keskiarvo:		79,5	160,9

Huomataan, että satunnaisesti generoidulla 100×100 -matriisilla on melko suuri kuntoluku. Kun parametri k on suuri, systeemissä esiintyy voimakasta värähtelyä, joka hidastaa DY-ratkaisijoita merkittävästi. Kuitenkin `ode45` ratkaisee DY-systeemin selvästi nopeammin kuin `ode15s`. Yleisesti `ode45` on kaikissa muissa tilanteissa `ode15s`:ää nopeampi ratkaisija paitsi jäykkien systeemien tapauksissa.

Olkoon C edelleen satunnaisesti generoitu 100×100 -matriisi. Valitaan nyt parametri k seuraavalla tavalla:

$$k = 100 \cdot \frac{1}{2\sigma_1^2},$$

missä σ_1 on C:n suurin singulaariarvo. Kerroin 100 on valittu sen takia, että valinnalla $k = \frac{1}{2\sigma_1^2}$ systeemi näyttäisi olevan niin jäykkä, ettei `ode45` pysty useimmissa tapauksissa ratkaisemaan probleemaa lainkaan. Kerroin suurentaa parametria k, jolloin systeemin jäykkyys vähenee. Nyt suurin osa matriisin A_c ominaisarvoista kuuluu $\pm 45°$ sektoriin reaaliakseliin nähden, mutta osa ominaisarvoista leviää negatiiviselle reaaliakselille, mikä selittää systeemin jäykkyyden. Asetetaan parametrille α edelleen heuristisesti saatu arvo 0,1. Lisäksi asetetaan DY-ratkaisijoille aikaraja $T_{\max} = 15 \text{ min} = 900 \text{ s}$. Jos tämä aikaraja ylittyy, keskeytetään systeemin ratkaisu ja todetaan, että ratkaisun hakemisessa meni liian kauan aikaa. Ajetaan MATLAB:issa simulaatioita näillä asetuksilla. Matriisin C kuntoluku $\text{cond}(C)$, parametri k ja laskenta-aika T DY-ratkaisijoille `ode45` ja `ode15s` on esitetty taulukossa 3.2.

Taulukko 3.2: Laskenta-aikoja C:n ollessa satunnaisesti generoitu 100×100 -matriisi ja systeemin ollessa jäykkä. Sarakkeissa on lueteltuna C:n kuntoluku, parametrin k arvo, ode45-ratkaisijan laskenta-aika sekunteina ja ode15s-ratkaisijan laskenta-aika sekunteina 10 eri matriisille. Laskenta-aika 900 s tarkoittaa, että ratkaisua ei ehditty hakea annetussa aikarajassa. Jäykälle systeemille ode15s on usein selvästi nopeampi ratkaisija.

$\text{cond}(C)$	k	T_{ode45} (s)	T_{ode15s} (s)
82,6	0,1341	9,0	6,4
90,9	0,1363	8,5	6,1
151,6	0,1347	33,4	11,9
243,5	0,1361	50,6	14,9
299,6	0,1309	492,0	61,1
338,4	0,1264	559,9	67,5
363,7	0,1343	468,1	66,2
942,1	0,1291	900,0	900,0
1068,3	0,1397	900,0	582,8
7916,8	0,1316	900,0	900,0
keskiarvo:		432,2	261,7

Huomataan, että parametrin k ollessa pieni laskenta-ajat ovat systeemin jäykkyydestä johtuen suuret, joskin `ode15s`-ratkaisija on toisinaan huomattavasti nopeampi kuin `ode45`. On järkevää, että `ode15s` on tässä tapauksessa nopeampi ratkaisija, sillä `ode15s` on suunniteltu nimenomaan jäykille systeemeille. Verrataan vielä taulukoiden 3.1 ja 3.2 tuloksia. Satunnaisesti generoidulla 100×100 -matriisilla kuntoluku voi vaihdella hyvinkin paljon. Kummassakin taulukossa on tapaus, jossa kuntoluvulle

pätee $\text{cond}(C) \approx 150$. Tällaisessa tapauksessa laskenta-ajat olivat seuraavanlaisia:

$$\begin{aligned} \text{pieni } k \quad &\Longrightarrow \quad T_{\texttt{ode45}} \approx 35 \text{ s}, \quad T_{\texttt{ode15s}} \approx 10 \text{ s}, \\ \text{suuri } k \quad &\Longrightarrow \quad T_{\texttt{ode45}} \approx 10 \text{ s}, \quad T_{\texttt{ode15s}} \approx 25 \text{ s}. \end{aligned}$$

Pienen k:n tapauksessa `ode15s` ratkaisi siis systeemin suunnilleen yhtä nopeasti kuin `ode45` suuren k:n tapauksessa. Kuitenkin `ode45` oli pienen k:n tapauksessa selvästi hitaampi kuin `ode15s` suuren k:n tapauksessa. Taulukossa 3.1 oli myös tapaus, jossa $\text{cond}(C) \approx 1200$, ja taulukossa 3.2 oli tapaus, jossa $\text{cond}(C) \approx 1100$. Huomataan, että kuntoluvun kasvaessa myös laskenta-aika kasvaa pääsääntöisesti. Edellä mainittujen kuntolukujen matriiseilla oli seuraavat laskenta-ajat:

$$\begin{aligned} \text{pieni } k \quad &\Longrightarrow \quad T_{\texttt{ode45}} > 900 \text{ s}, \quad T_{\texttt{ode15s}} \approx 600 \text{ s}, \\ \text{suuri } k \quad &\Longrightarrow \quad T_{\texttt{ode45}} \approx 250 \text{ s}, \quad T_{\texttt{ode15s}} \approx 500 \text{ s}. \end{aligned}$$

Kummassakin tapauksessa laskenta-ajat olivat pitkiä, mutta parametrin k ollessa suuri sekä `ode45` että `ode15s` ratkaisivat systeemin nopeammin kuin pienen k:n tapauksessa.

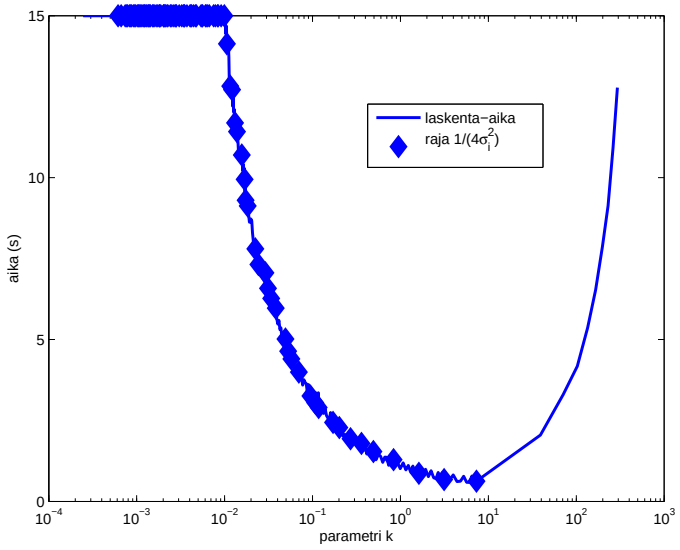
Annetaan seuraavaksi parametrin k saada reaalilukuarvoja väliltä

$$\frac{1}{10\sigma_1^2} \leq k \leq \frac{10}{\sigma_p^2},$$

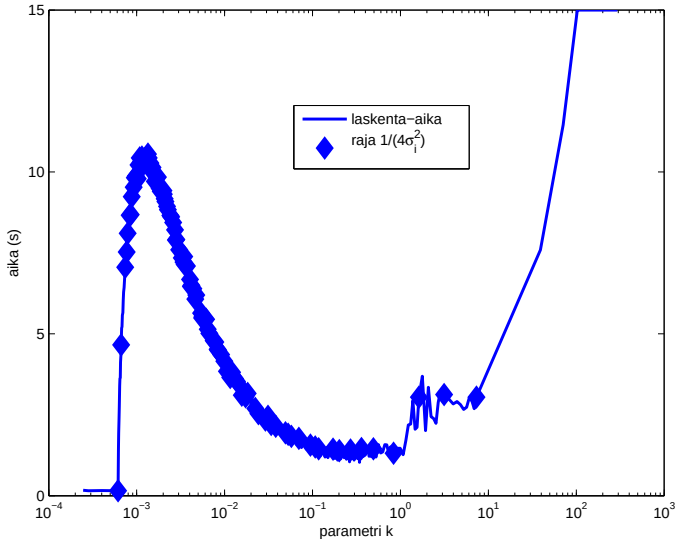
ratkaistaan differentiaaliyhtälö (3.3) ratkaisijoilla `ode45` ja `ode15s` ja kirjataan laskentaan käytetty aika. Koska laskenta-ajat voivat vaihdella hyvin paljon lausekkeesta $\frac{1}{\sigma_i^2}$ riippuen, asetetaan laskennan nopeuttamiseksi ratkaisijoille aikaraja kullakin k:n arvolla. Jos ratkaisua ei ole saavutettu 15 sekunnin kuluessa, todetaan parametrin k arvo laskennallisesti tehottomaksi ja siirrytään seuraavaan k:n arvoon. Tarkastellaan laskenta-aikoja C:n ollessa ensin 100×100 -neliömatriisi ja sitten 150×350 -matriisi. Ratkaisijoilla `ode45` ja `ode15s` saadut laskenta-ajat neliömatriisin tapauksessa on esitetty kuvissa 3.7 ja 3.8, vastaavasti. Leveän matriisin tapauksessa `ode45`:llä ja `ode15s`:llä saadut laskenta-ajat on esitetty kuvissa 3.9 ja 3.10, vastaavasti. Kuvissa 3.7–3.10 on laskenta-ajan lisäksi merkitty samaan kuvaajaan ne parametrin k arvot, joilla matriisin A_c ominaisarvoihin tulee aina yksi kompleksilukupari enemmän. Näin käy, kun $k = \frac{1}{4\sigma_i^2}$.

Valitaan jatkossa parametri k mahdollisimman suureksi, jotta `ode45`-ratkaisija toimisi mahdollisimman tehokkaasti. Kuvan 3.7 perusteella valitaan k lausekkeiden $\frac{1}{4\sigma_{p-1}^2}$ ja $\frac{1}{4\sigma_p^2}$ keskiarvoksi:

$$k = \frac{1}{2}\left(\frac{1}{4\sigma_{p-1}^2} + \frac{1}{4\sigma_p^2}\right) = \frac{1}{8}\left(\frac{1}{\sigma_{p-1}^2} + \frac{1}{\sigma_p^2}\right). \tag{3.25}$$



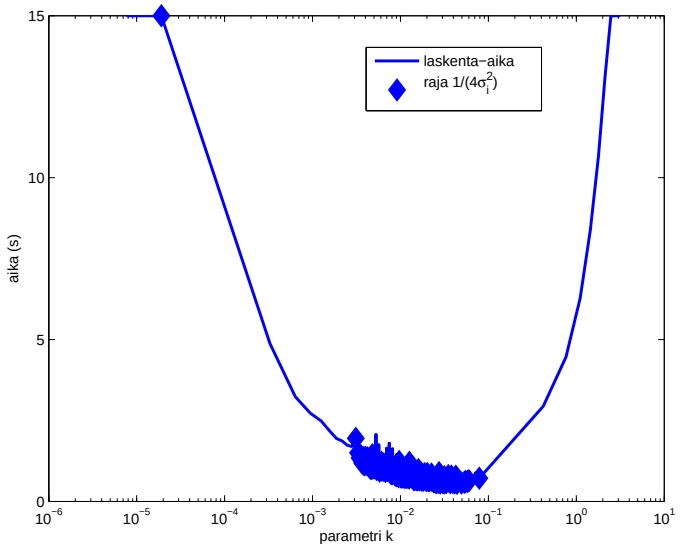
Kuva 3.7: Ratkaisijan ode45 laskenta-ajat parametrin k eri arvoilla, kun C on kokoa 100×100. Samaan kuvaajaan on merkitty timantti $\diamond$ niiden k:n arvojen kohtaan, joilla matriisin A_c ominaisarvoihin tulee yksi kompleksilukupari enemmän. Uusi kompleksinen juuripari muodostuu aina rajalla $k = \frac{1}{4\sigma_i^2}$, missä $i = 1, 2, \ldots, p$. Laskenta-aika on lyhimmillään, kun $\frac{1}{4\sigma_{p-1}^2} \leq k \leq \frac{1}{4\sigma_p^2}$. Huomaa, että x-akseli on logaritmisesti skaalattu.



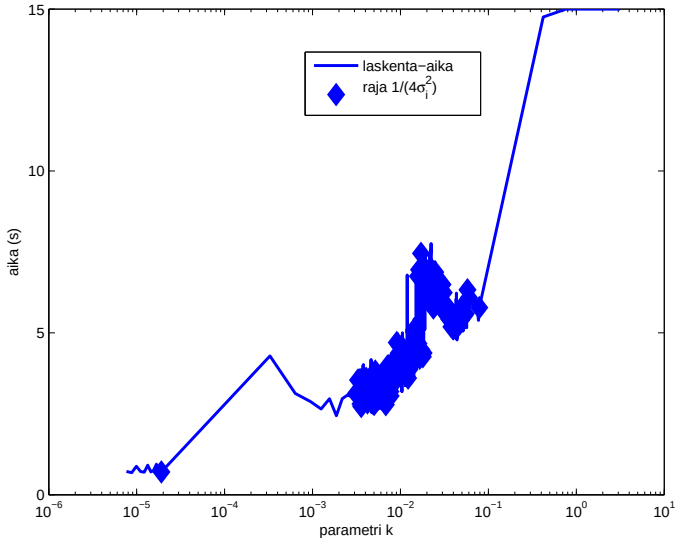
Kuva 3.8: Ratkaisijan ode15s laskenta-ajat parametrin k eri arvoilla, kun C on kokoa 100×100. Simulaatiossa on käytetty samaa matriisia ja samaa systeemiä kuin kuvan 3.7 simulaatiossa. Samaan kuvaajaan on merkitty timanteilla $\diamond$ ne k:n arvot, joilla matriisin A_c ominaisarvoihin tulee yksi kompleksilukupari enemmän. Laskenta-aika on lyhimmillään, kun $k \leq \frac{1}{4\sigma_1^2}$. Huomaa, että x-akseli on logaritmisesti skaalattu.

Vastaavasti kuvan 3.8 perusteella valitaan `ode15s`-ratkaisijalle parametrin k arvoksi

$$k = \frac{1}{4\sigma_1^2}, \tag{3.26}$$



Kuva 3.9: Ratkaisijan ode45 laskenta-ajat parametrin k eri arvoilla, kun C on kokoa 150×350. Samaan kuvaajaan on merkitty timanteilla $\diamond$ ne k:n arvot, joilla matriisin A_c ominaisarvoihin tulee yksi kompleksilukupari enemmän. Laskenta-aika on lyhimmillään, kun $\frac{1}{4\sigma_{144}^2} \leq k \leq \frac{1}{4\sigma_{145}^2}$. Huomaa, että x-akseli on logaritmisesti skaalattu.



Kuva 3.10: Ratkaisijan ode15s laskenta-ajat parametrin k eri arvoilla, kun C on kokoa 150×350. Simulaatiossa on käytetty samaa matriisia ja samaa systeemiä kuin kuvan 3.9 simulaatiossa. Samaan kuvaajaan on merkitty timanteilla $\diamond$ ne k:n arvot, joilla matriisin A_c ominaisarvoihin tulee yksi kompleksilukupari enemmän. Laskenta-aika on lyhimmillään, kun $k \leq \frac{1}{4\sigma_1^2}$. Huomaa, että x-akseli on logaritmisesti skaalattu.

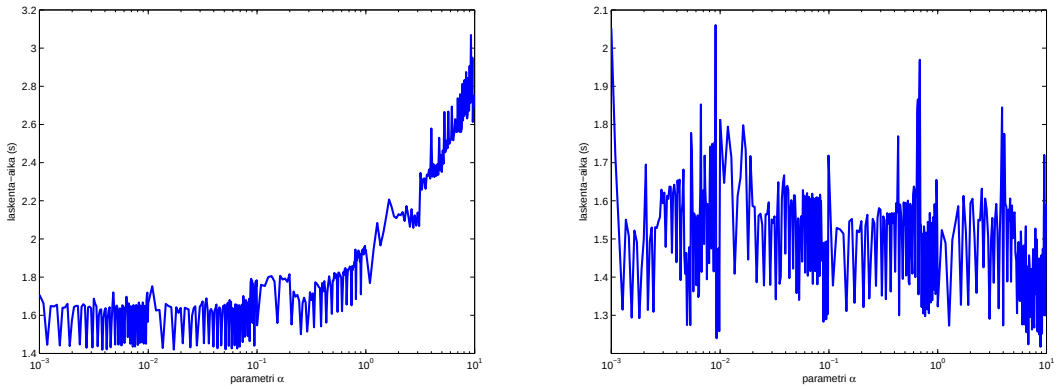
koska pienillä k:n arvoilla aaltoilu on pientä ja koska ratkaisija on tarkoitettu jäykille systeemeille.

3.4.2 Parametri α

Selvitetään seuraavaksi, miten parametri α kannattaa valita. Tähän mennessä α:lle on käytetty kokeilemalla saatua arvoa 0,1. Annetaan α:n saada reaalilukuarvoja väliltä

$$\frac{1}{1000} \leq \alpha \leq 10$$

ja ratkaistaan differentiaaliyhtälö ratkaisijoilla `ode45` ja `ode15s`. Valitaan $\widetilde{C}^* = C^*$ ja asetetaan parametri k kaavan (3.25) mukaisesti `ode45`-ratkaisijalle, ja kaavan (3.26) mukaisesti `ode15s`-ratkaisijalle. Kirjataan laskentaan käytetty aika. Simulaatioissa havaitaan, ettei parametrin α valinta ole laskenta-ajan kannalta läheskään yhtä merkittävä kuin parametrin k valinta. Laskenta-ajat eri α:n arvoilla on piirretty `ode45`-ratkaisijalle kuvaan 3.11 ja `ode15s`-ratkaisijalle kuvaan 3.12.



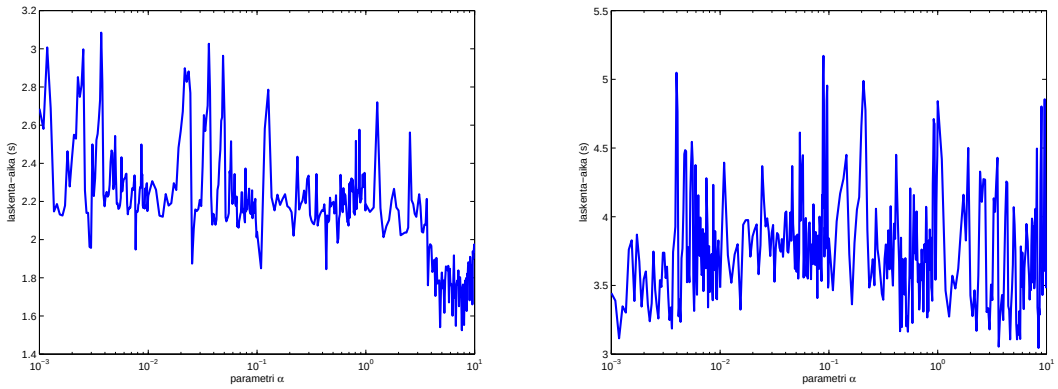
Kuva 3.11: Laskenta-ajat eri α:n arvoilla, kun differentiaaliyhtälön ratkaisemiseen käytetään ode45:tä. Vasemmassa kuvaajassa matriisi C on kokoa 100×100, oikeassa kuvaajassa C on kokoa 500×1250. Neliömatriisin tapauksessa laskenta-aika on pienimmillään, kun $\alpha \leq 0{,}1$. Leveän matriisin tapauksessa laskenta-aikojen vaihtelut ovat melko pieniä. Huomaa, että x-akseli on kummassakin kuvaajassa skaalattu logaritmisesti.

Kuvien 3.11 ja 3.12 perusteella valitaan jatkossa parametri α seuraavalla tavalla:

$$\alpha_{\text{ode45}} = \frac{1}{100} \qquad \text{ja} \qquad \alpha_{\text{ode15s}} = 10. \tag{3.27}$$

Parametrin valinnassa on painotettu sitä, että α vaikuttaa laskenta-aikoihin enemmän neliömatriiseilla kuin leveillä matriiseilla. Samat α:n arvot toimivat hyvin myös muilla parametrien $\widetilde{C}^*$ ja k valinnoilla, koska matriisin A_c ominaisarvojen lausekkeessa (3.13) α on vain skaalauskerroin, joka ei riipu parametreista $\widetilde{C}^*$ ja k.

Muista differentiaaliyhtälöiden ratkaisijoista `ode23` ja `ode113` on tarkoitettu epäjäykille systeemeille, ja ne toimivat tehokkaasti, kun parametrit α ja k valitaan samoin kuin `ode45`:lle. Ratkaisijat `ode23s`, `ode23t` ja `ode23tb` on tarkoitettu jäykille systeemeille, ja vastaavasti ne toimivat tehokkaasti, kun α ja k valitaan kuten `ode15s`:lle.



Kuva 3.12: Laskenta-ajat eri α:n arvoilla, kun differentiaaliyhtälön ratkaisemiseen käytetään ode15s:ää. Vasemmassa kuvaajassa matriisi C on kokoa 300×300, oikeassa kuvaajassa C on kokoa 300×750. Matriisit valittiin ode15s:lle eri kokoisiksi kuin ode45:lle (kuva 3.11), koska ratkaisijoista ode15s oli selvästi nopeampi neliömatriisien tapauksessa ja selvästi hitaampi leveillä matriiseilla. Neliömatriisilla laskenta-aika on pienimmillään, kun α on mahdollisimman suuri. Leveällä matriisilla laskenta-aikojen vaihtelut ovat melko pieniä. Huomaa, että x-akseli on kummassakin kuvaajassa skaalattu logaritmisesti.

3.5 Singulaariarvohajotelman hyödyntäminen differentiaaliyhtälön ratkaisemiseksi

Tarkastellaan vaihtoehtoista lähestymistapaa, jolla matriisin A_c ominaisarvot saadaan haluttuun paikkaan kompleksitasossa riippumatta matriisin C singulaariarvoista. Aikaisemmin ominaisarvoille johdettiin seuraava lauseke (kaava (3.13)):

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i^2}\right), \qquad i = 1, 2, \ldots, p.$$

Tarkkaan ottaen nämä $2p$ ominaisarvoa ovat muotoa

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i\tilde{\sigma}_i}\right), \qquad i = 1, 2, \ldots, p, \tag{3.28}$$

missä σ_i on C:on singulaariarvo ja $\tilde{\sigma}_i$ on parametrina toimivan matriisin $\widetilde{C}^*$:n singulaariarvo. Aiemmin valittiin $\widetilde{C}^* = C^*$. Matriisilla ja sen konjugaattitranspoosilla on samat singulaariarvot, joten tässä tapauksessa oli $\tilde{\sigma}_i = \sigma_i$. Entä jos parametri $\widetilde{C}^*$ valittaisiinkin jollain toisella tavalla? Esitellään kaksi vaihtoehtoista tapaa valita $\widetilde{C}^*$. Kumpikin menetelmä vaikuttaa siihen, miten parametri k kannattaa valita.

3.5.1 Parametrina pseudoinverssi

Määritelmän 2.1.17 nojalla matriisin C pseudoinverssi $C^+ = V\Lambda^+U^*$ on kooltaan $n \times p$, ja Λ^+:n diagonaalialkiot ovat $\frac{1}{\sigma_1}$, $\frac{1}{\sigma_2}$, …, $\frac{1}{\sigma_p}$. Kun valitaan $\widetilde{C}^* = C^+$, saadaan matriisin $\widetilde{C}^*$ singulaariarvoiksi $\tilde{\sigma}_i = \frac{1}{\sigma_i}$. Tällöin yhtälöstä (3.28) saadaan matriisin

A_c ominaisarvoiksi:

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i \frac{1}{\sigma_i}}\right) = \frac{\alpha}{2}\left(-1 \pm \sqrt{1-4k}\right), \quad i = 1, 2, \ldots, p. \tag{3.29}$$

Näin p kappaletta A_c:n ominaisarvoista saadaan asetettua samaan pisteeseen $-\frac{\alpha}{2} + \frac{\alpha}{2}\sqrt{1-4k}$, ja toiset p kappaletta pisteeseen $-\frac{\alpha}{2} - \frac{\alpha}{2}\sqrt{1-4k}$. Loput $n-p$ ominaisarvoa sijaitsevat luvun 3.2 sivun 36 tuloksen perusteella pisteessä $-\alpha$.

Huomataan, että ominaisarvot ovat reaalisia, kun $1 - 4k \geq 0$, eli

$$k \leq \frac{1}{4}.$$

Jos halutaan, että ominaisarvot ovat kompleksisia ja sijaitsevat $\pm 45°$ sektorissa reaaliakseliin nähden, on oltava

$$\frac{1}{4} < k \leq \frac{1}{2}.$$

Esimerkiksi kun $k = \frac{1}{2}$, ominaisarvoille on voimassa

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{-1}\right) = \frac{\alpha}{2}\left(-1 \pm \mathrm{i}\right),$$

jolloin $|\mathrm{Re}(\lambda_i)| = |\mathrm{Im}(\lambda_i)| = \frac{\alpha}{2}$, ja ominaisarvot sijaitsevat $\pm 45°$ kulmassa reaaliakselista katsottuna, $i = 1, 2, \ldots, p$.

Ratkaistaan differentiaaliyhtälö (3.3) eri parametrin k arvoilla, kun

$$A_c = \begin{bmatrix} -\alpha I_n & -k\alpha^2 C^+ \\ C & O \end{bmatrix}, \tag{3.30}$$

parametri α valitaan kaavan (3.27) mukaisesti ja k saa arvot

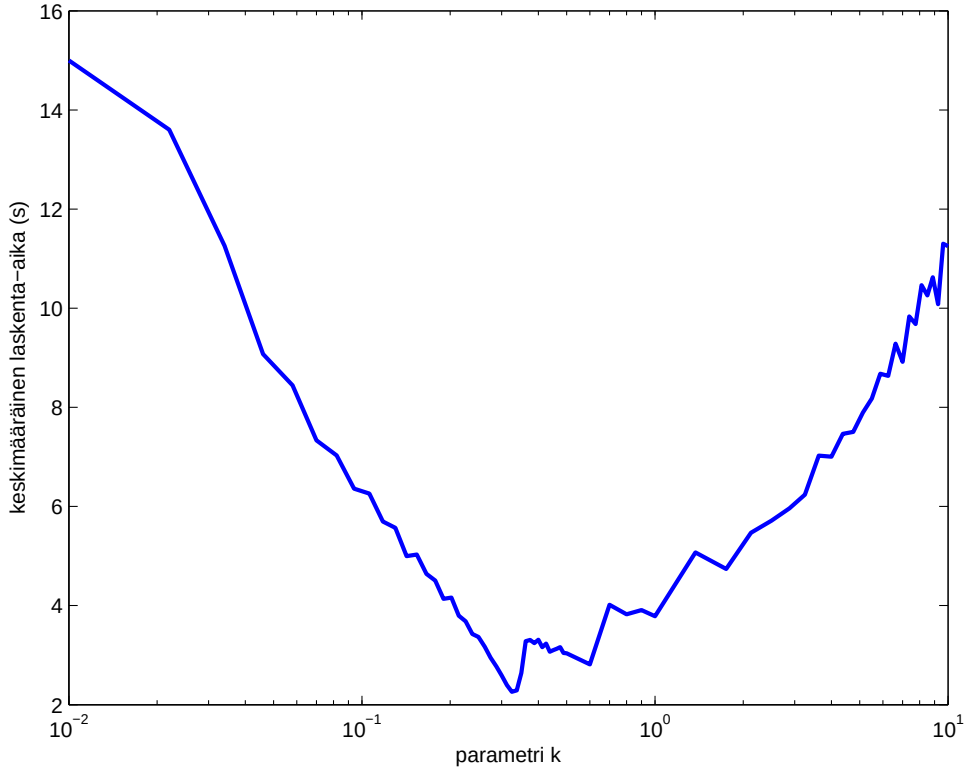
$$\frac{1}{100} \leq k \leq 10.$$

Ajetaan simulaatio sadalla eri 750×750 -neliömatriisilla ja lasketaan ratkaisijoiden keskimääräiset laskenta-ajat kullakin k:n arvolla. Ajan säästämiseksi asetetaan `ode45`:lle aikarajaksi 15 sekuntia jokaisella yksittäisellä k:n arvolla. Ratkaisijan `ode45` laskenta-ajat on esitetty kuvassa 3.13, ja ratkaisijan `ode15s` laskenta-ajat kuvassa 3.14.

Tutkitaan vielä simulaatioiden avulla, muuttuuko laskenta-aikojen kuvaajan muoto merkittävästi, kun matriisi onkin leveä. Valitaan edelleen parametri k väliltä

$$\frac{1}{100} \leq k \leq 10,$$

parametri α kaavan (3.27) mukaan ja matriisi A_c siten, että se on muotoa (3.30).



Kuva 3.13: Ratkaisijan ode45 keskimääräiset laskenta-ajat 750×750 -matriisille C parametrin k eri arvoilla, kun differentiaaliyhtälön ratkaisussa hyödynnetään C:n pseudoinverssiä ($\widetilde{C}^* = C^+$). Kullakin k:n arvolla laskenta-aika on saatu 100 simulaation keskiarvona. 15 sekunnin laskenta-aika tarkoittaa, ettei systeemiä saatu ratkaistua 15 sekunnin aikarajaan mennessä. Keskimääräinen laskenta-aika on pienimmillään, kun $0{,}30 \leq k \leq 0{,}35$. Huomaa, että x-akseli on logaritmisesti skaalattu.

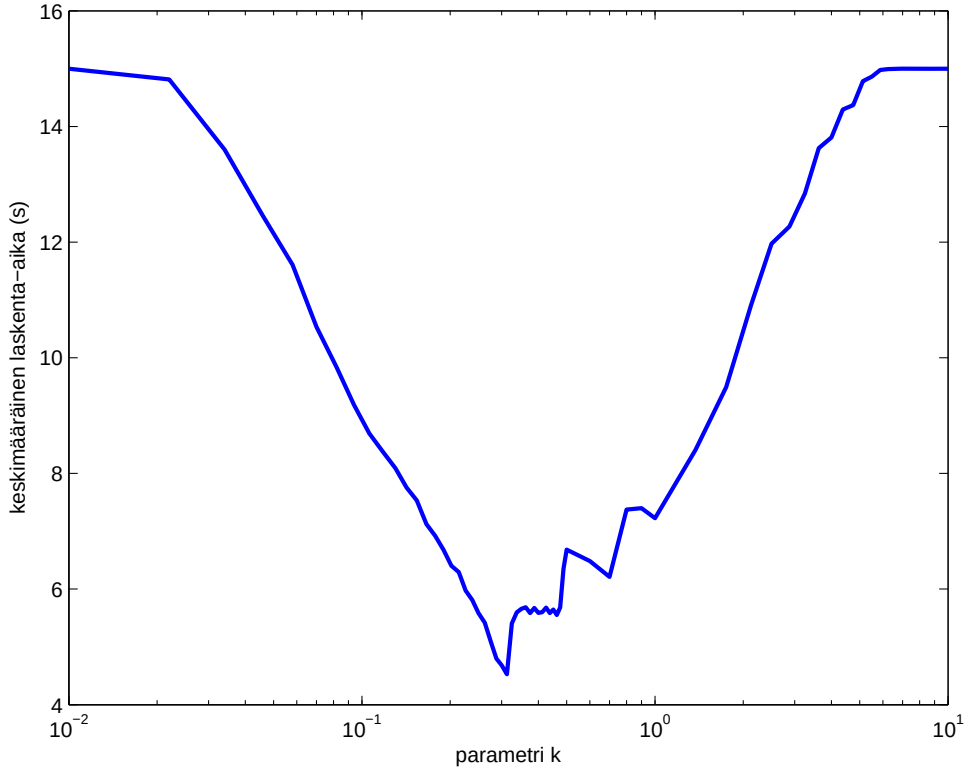
Ratkaistaan sitten differentiaaliyhtälö (3.30) sadalla eri 500×1250 -matriisilla käyttäen ratkaisijoita `ode45` ja `ode15s`. Asetetaan jälleen yksittäiseen k:n arvoon liittyväksi maksimilaskenta-ajaksi 15 sekuntia. Ratkaisijoiden `ode45` ja `ode15s` keskimääräiset laskenta-ajat on esitetty kuvassa 3.15.

Huomataan, että matriisin koosta sekä ratkaisijasta riippumatta laskenta-ajat ovat pienimmillään, kun $0{,}30 \leq k \leq 0{,}35$. Valitaan siis

$$k = 0{,}325. \tag{3.31}$$

Tällöin matriisin A_c ominaisarvoille λ_i on voimassa kaikilla $i = 1, 2, \ldots, p$:

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4 \cdot 0{,}325}\right) = \frac{\alpha}{2}\left(-1 \pm \mathrm{i}\sqrt{0{,}3}\right),$$

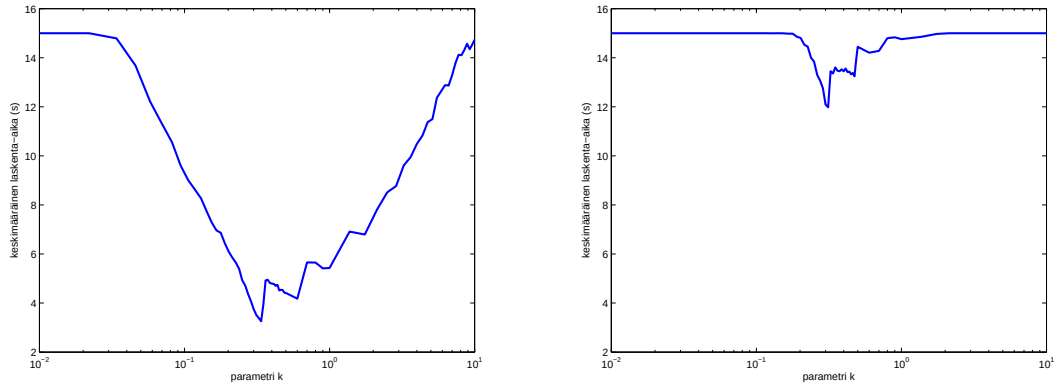


Kuva 3.14: Ratkaisijan ode15s keskimääräiset laskenta-ajat 750×750 -matriisille C parametrin k eri arvoilla, kun differentiaaliyhtälön ratkaisussa hyödynnetään C:n pseudoinverssiä ($\widetilde{C}^* = C^+$). Kullakin k:n arvolla laskenta-aika on saatu 100 simulaation keskiarvona. 15 sekunnin laskenta-aika tarkoittaa, ettei systeemiä saatu ratkaistua 15 sekunnin aikarajaan mennessä. Kuten ode45-ratkaisijalla, keskimääräinen laskenta-aika on pienimmillään myös ode15s:llä, kun $0{,}30 \leq k \leq 0{,}35$. Huomaa, että x-akseli on logaritmisesti skaalattu.

ja kulma reaaliakseliin nähden on

$$\begin{aligned} |\theta| &= \left|\arctan\left(\frac{\mathrm{Im}(\lambda_i)}{\mathrm{Re}(\lambda_i)}\right)\right| \\ &= \left|\arctan\left(\frac{\pm\sqrt{0{,}3}\cdot\alpha/2}{-\alpha/2}\right)\right| \\ &= \left|\arctan(\sqrt{0{,}3})\right| \\ &\approx 28{,}71^\circ. \end{aligned} \tag{3.32}$$

Ominaisarvot ovat siis $\pm 28{,}71^\circ$ kulmassa reaaliakseliin nähden. Tällöin ne kuuluvat $\pm 45^\circ$ sektoriin, eikä systeemi myöskään ole jäykkä.



Kuva 3.15: Vasemmassa kuvaajassa on ratkaisijan ode45 keskimääräiset laskenta-ajat 500×1250 -matriisille C parametrin k eri arvoilla, ja oikeassa kuvaajassa vastaavasti ode15s:n laskenta-ajat, kun differentiaaliyhtälön ratkaisussa hyödynnetään C:n pseudoinverssiä ($\widetilde{C}^* = C^+$). Kullakin k:n arvolla laskenta-aika on saatu 100 simulaation keskiarvona. 15 sekunnin laskenta-aika tarkoittaa, ettei systeemiä saatu ratkaistua 15 sekunnin aikarajaan mennessä. Kuten oikeasta kuvaajasta huomataan, pseudoinverssin käyttö on melko tehotonta ode15s:llä, kun C on leveä matriisi. Kummallakin ratkaisijalla keskimääräinen laskenta-aika on pienimmillään, kun $0{,}30 \leq k \leq 0{,}35$. Huomaa, että x-akseli on logaritmisesti skaalattu.

3.5.2 Parametrina skaalausmatriisi

Tarkastellaan seuraavaksi differentiaaliyhtälöä (3.3), kun

$$A_c = \begin{bmatrix} -\alpha I_n & -k\alpha^2 \widetilde{C}^* \\ C & O \end{bmatrix}$$

ja $\widetilde{C}^*$ valitaan singulaariarvohajotelman avulla siten, että C:n singulaariarvot eivät supistukaan. Kaavan (3.28) perusteella matriisin A_c ominaisarvot ovat muotoa

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i\tilde{\sigma}_i}\right), \qquad i = 1, 2, \ldots, p,$$

missä σ_i on C:n singulaariarvo ja $\tilde{\sigma}_i$ matriisin $\widetilde{C}^*$ singulaariarvo. Luvussa 3.3 näytettiin, että `ode45` ratkaisee differentiyhtälösysteemejä huomattavasti tehokkaammin C:n ollessa leveä matriisi verrattuna tapaukseen, jossa C on neliömatriisi. Ilmiön selityksenä oli lyhyesti sanottuna se, että leveän matriisin singulaariarvot ovat suurempia kuin neliömatriisin. Tällöin yllä olevassa yhtälössä myös lauseke $\sigma_i\tilde{\sigma}_i$ on suurempi leveillä matriiseilla kuin neliömatriiseilla, jolloin A_c:n ominaisarvojen reaaliosat pysyvät kaukana imaginaariakselilta ja ominaisarvojen imaginaariosat ovat pieniä. Pyritään jäljittelemään ilmiötä neliömatriisien tapauksessa kasvattamalla lausekkeen $\sigma_i\tilde{\sigma}_i$ arvoa. Lauseketta halutaan kasvattaa erityisesti pienimpien singulaariarvojen kohdalla. Oletetaan, että C:n pienimmät singulaariarvot ovat σ_m, σ_{m+1}, $\ldots$, σ_p, ja lauseketta $\sigma_i\tilde{\sigma}_i$ halutaan kasvattaa juuri näillä indekseillä, $m \leq i \leq p$. Jos

valitaan

$$\tilde{\sigma}_i = \frac{a^2}{\sigma_i}, \quad i = m, m+1, \ldots, p, \tag{3.33}$$

niin lauseke $\sigma_i\tilde{\sigma}_i$ saadaan asetettua arvoon

$$\sigma_i\tilde{\sigma}_i = \sigma_i\frac{a^2}{\sigma_i} = a^2.$$

Jos valitaan reaaliluku a väliltä $\sigma_m < a \leq \sigma_{m-1}$, kaikki lausekkeet $\sigma_i\tilde{\sigma}_i$, $m \leq i \leq p$, saadaan nostettua arvoon a^2. Muille singulaariarvoille voidaan valita $\tilde{\sigma}_i = \sigma_i$, jolloin

$$\sigma_i\tilde{\sigma}_i = \begin{cases} \sigma_i^2, & \text{kun } 1 \leq i \leq m-1 \\ a^2, & \text{kun } m \leq i \leq p \end{cases}.$$

Jotta lauseke $\sigma_i\tilde{\sigma}_i$ olisi mahdollisimman suuri, kun $m \leq i \leq p$, valitaan $a = \sigma_{m-1}$, jolloin kaikki $(p - m + 1)$ pienintä lauseketta skaalautuvat arvoon σ_{m-1}^2.

Simulaatioissa osoittautuu, että laskenta on tehokkainta, kun $m = 2$ ja valitaan $a = \sigma_1$. Tällöin $\sigma_i\tilde{\sigma}_i = \sigma_1^2$, kun $2 \leq i \leq p$. Tällöin kaikki paitsi suurinta singulaariarvoa vastaava lauseke on skaalattu suurimman singulaariarvon mukaan. Kun lisäksi valitaan $\tilde{\sigma}_1 = \sigma_1$, kaikille lausekkeille pätee $\sigma_i\tilde{\sigma}_i = \sigma_1^2$, $i = 1, 2, \ldots, p$. Seuraava kysymys on, miten matriisin $\widetilde{C}^*$ singulaariarvot saadaan kaavan (3.33) mukaiseen muotoon. Tiedetään, että singulaariarvohajotelman avulla C voidaan esittää muodossa $C = U\Lambda V^*$. Muodostetaan $\widetilde{C}^*$:lle Λ^+:a vastaava digonaalimatriisi, joka skaalaa lausekkeet $\sigma_i\tilde{\sigma}_i$ arvoon σ_1^2:

$$\Lambda^{\sigma_1^2} = \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 \\ 0 & \sigma_1^2/\sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_1^2/\sigma_p \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}. \tag{3.34}$$

Skaalausmatriisi $\Lambda^{\sigma_1^2}$ on siis kokoa $n \times p$, $n \geq p$. Kun nyt valitaan

$$\widetilde{C}^* = V\Lambda^{\sigma_1^2}U^* =: C^{\sigma_1^2}, \tag{3.35}$$

matriisin A_c ominaisarvoiksi saadaan

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i\tilde{\sigma}_i}\right) = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_1^2}\right), \qquad i = 1, 2, \ldots, p.$$

Näin pienten singulaariarvojen vaikutus saadaan eliminoitua. Matriisia $C^{\sigma_1^2}$ sano-

taan C:n skaalausmatriisiksi. Kun huomataan, että $\Lambda^{\sigma_1^2} = \sigma_1^2\Lambda^+$, saadaan skaalausmatriisin ja pseudoinverssin välille seuraava yhteys:

$$C^{\sigma_1^2} = V\Lambda^{\sigma_1^2}U^* = V\sigma_1^2\Lambda^+U^* = \sigma_1^2(V\Lambda^+U^*) = \sigma_1^2C^+.$$

Parametrin k valinnasta tiedetään, että matriisin A_c ominaisarvot ovat kompleksisia, kun

$$k > \frac{1}{4\sigma_1^2} = \frac{1/4}{\sigma_1^2}.$$

Simulaatioiden avulla havaitaan, että laskenta-ajat ovat pienimmillään, kun parametri k saa arvot

$$\frac{3/10}{\sigma_1^2} \leq k \leq \frac{6/10}{\sigma_1^2}. \tag{3.36}$$

Kun k valitaan yhtälön (3.36) alarajan mukaan, A_c:n ominaisarvot ovat kaikilla $i = 1, 2, \ldots, p$

$$\begin{aligned}\lambda_i &= \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4\frac{3/10}{\sigma_1^2}\sigma_1^2}\right)\\ &= \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 12/10}\right)\\ &= \frac{\alpha}{2}\left(-1 \pm \mathrm{i}\sqrt{1/5}\right),\end{aligned}$$

ja kulma reaaliakseliin nähden on

$$\begin{aligned}|\theta| &= \left|\arctan\left(\frac{\mathrm{Im}(\lambda_i)}{\mathrm{Re}(\lambda_i)}\right)\right|\\ &= \left|\arctan\left(\frac{\pm 1/\sqrt{5}\cdot\alpha/2}{-\alpha/2}\right)\right|\\ &= \left|\arctan(1/\sqrt{5})\right|\\ &\approx 24{,}09^\circ.\end{aligned} \tag{3.37}$$

Ominaisarvot ovat nyt siis $\pm 24{,}09^\circ$ kulmassa reaaliakseliin nähden. Ne kuuluvat $\pm 45^\circ$ sektoriin, eikä systeemi myöskään ole jäykkä. Jos taas k valitaan yhtälön (3.36) ylärajan mukaan, A_c:n ominaisarvot ovat kaikilla $i = 1, 2, \ldots, p$

$$\begin{aligned}\lambda_i &= \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4\frac{6/10}{\sigma_1^2}\sigma_1^2}\right)\\ &= \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 12/5}\right)\\ &= \frac{\alpha}{2}\left(-1 \pm \mathrm{i}\sqrt{7/5}\right),\end{aligned}$$

ja kulma reaaliakseliin nähden on

$$\begin{aligned} |\theta| &= \left|\arctan\left(\frac{\mathrm{Im}(\lambda_i)}{\mathrm{Re}(\lambda_i)}\right)\right| \\ &= \left|\arctan\left(\frac{\pm\sqrt{7/5}\cdot\alpha/2}{-\alpha/2}\right)\right| \\ &= \left|\arctan(\sqrt{7/5})\right| \\ &\approx 49{,}80^\circ. \end{aligned} \tag{3.38}$$

Nyt ominaisarvot ovat siis $\pm 49{,}80^\circ$ kulmassa reaaliakseliin nähden, eli hieman $\pm 45^\circ$ sektorin ulkopuolella. Systeemi ei ole jäykkä. Simulaatioiden perusteella valitaan k ratkaisijoille `ode45` ja `ode15s` seuraavalla tavalla:

$$k_{\mathrm{ode45}} = \frac{0{,}475}{\sigma_1^2} \quad \text{ja} \quad k_{\mathrm{ode15s}} = \frac{0{,}375}{\sigma_1^2}. \tag{3.39}$$

Tällöin ominaisarvojen kulmat reaaliakseliin nähden ovat

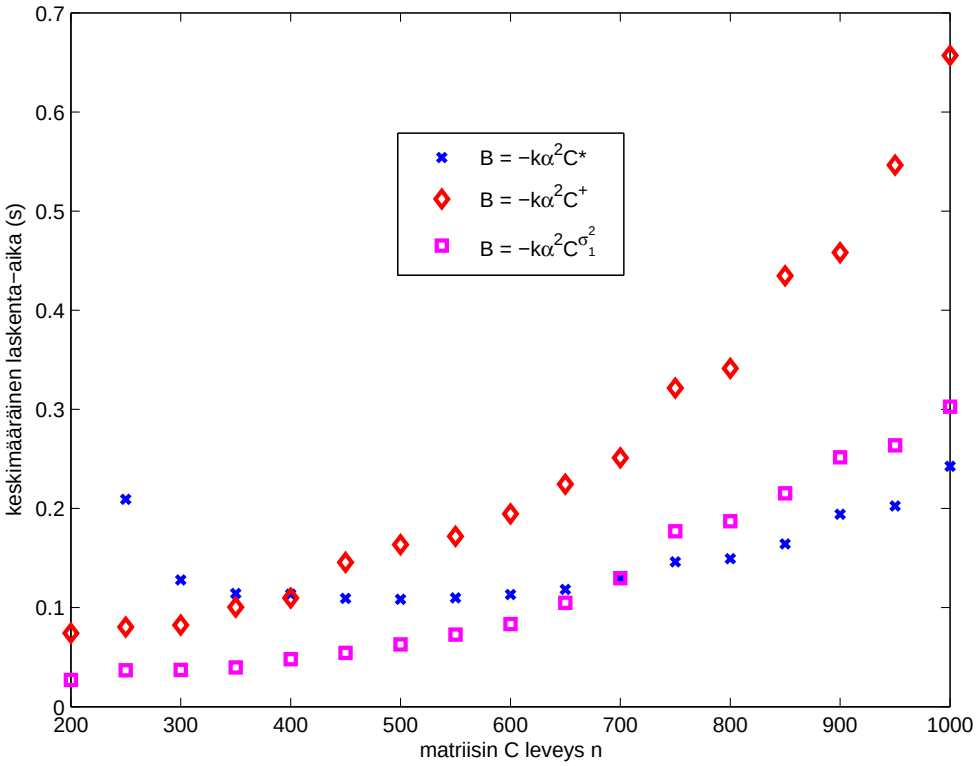
$$\theta_{\mathrm{ode45}} \approx \pm 43{,}49^\circ \quad \text{ja} \quad \theta_{\mathrm{ode15s}} \approx \pm 35{,}26^\circ. \tag{3.40}$$

3.5.3 Singulaariarvohajotelman kannattavuus eri kokoisilla matriiseilla

Mikäli tarkoituksena olisi ratkaista vain yksittäinen yhtälöryhmä $C\mathbf{x} = \mathbf{r}$, ratkaisu voitaisiin muodostaa jo C:n singulaariarvohajotelman avulla, sillä kappaleen 2.1.3 perusteella yhtälöryhmän eräs ratkaisu on $\mathbf{x} = C^+\mathbf{r}$. Tutkimuskysymyksen kannalta on kuitenkin oleellisesta ratkaista yhtälöryhmä $C_i\mathbf{x}_i = \mathbf{r}_i$ toistuvasti, $i = 1, 2, \ldots, N$. Siksi singulaariarvohajotelman tarkoituksena ei ole suoraan ratkaista yhtälöryhmää vaan nopeuttaa DY-ratkaisijaa.

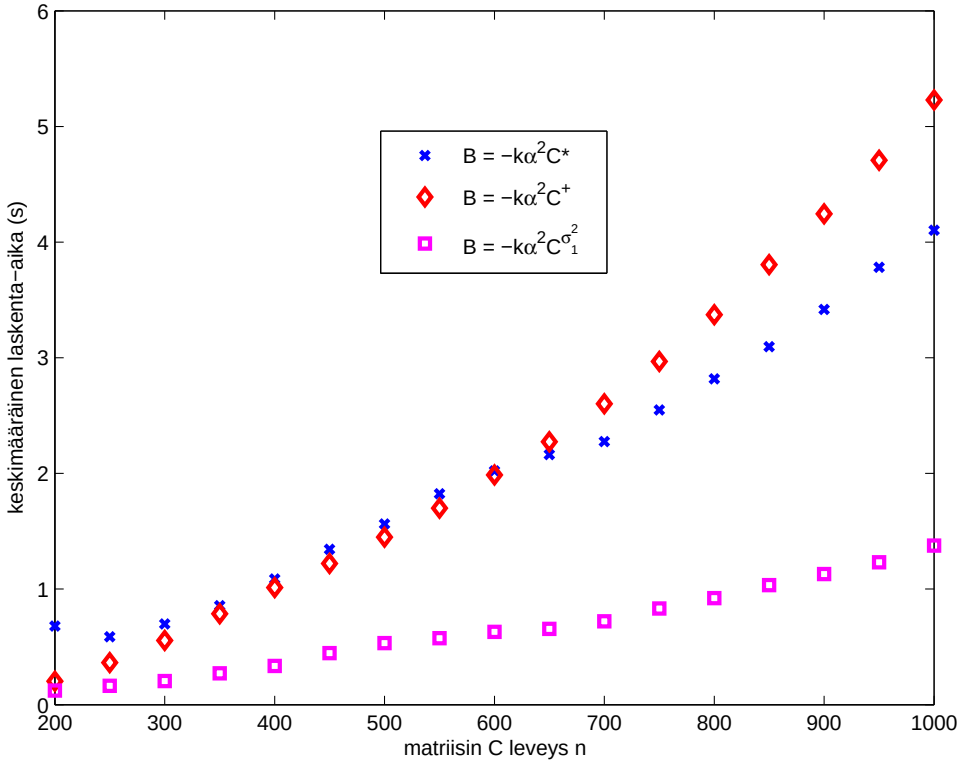
Kuvissa 3.13–3.15 tehdyissä simulaatioista huomattiin, että singulaariarvohajotelma näyttäisi nopeuttavan differentiaaliyhtälön ratkaisua enemmän neliömatriiseilla kuin leveillä matriiseilla. Tutkitaan, miten singulaariarvohajotelmaa hyödyntäessä ratkaisijoiden laskenta-ajat muuttuvat, kun matriisin C leveyttä kasvatetaan. Tarkastellaan jälleen $p \times n$ -matriiseja C, joilla $p = 200$ ja n saa arvot $n_1 = 200$, $n_2 = 250$, …, $n_{17} = 1000$. Käydään erikseen läpi tapaukset $\widetilde{C}^* = C^+$ ja $\widetilde{C}^* = C^{\sigma_1^2}$. Valitaan parametrin k arvo C^+:lle kaavan (3.31) mukaan ja $C^{\sigma_1^2}$:lle kaavan (3.39) mukaan. Ratkaistaan differentiaaliyhtälö sekä `ode45`:llä että `ode15s`:llä. Asetetaan parametri α kaavan (3.27) mukaisesti: `ode45`:lle $\frac{1}{100}$ ja `ode15s`:lle 10. Jokaista matriisikokoa $p \times n_i$ kohti generoidaan 1000 erilaista matriisia, ja kyseisen koon laskenta-aika saadaan laskenta-aikojen keskiarvona. Verrataan saatuja tuloksia aiempiin ku-

vien 3.5 ja 3.6 tuloksiin, joissa ei hyödynnetty singulaariarvohajotelmaa vaan käytettiin konjugaattitranspoosia. Aiemmissa luvuissa siis matriisille $B = -k\alpha^2\widetilde{C}^*$ valittiin $\widetilde{C}^* = C^*$, mutta singulaariarvohajotelmaa käytettäessä valitaan joko $\widetilde{C}^* = C^+$ tai $\widetilde{C}^* = C^{\sigma_1^2}$. Matriisille $\widetilde{C}^*$ on siis yhteensä 3 vaihtoehtoa: konjugaattitranspoosi C^*, pseudoinverssi C^+ ja skaalausmatriisi $C^{\sigma_1^2}$. Näistä kaksi jälkimmäistä käyttävät singulaariarvohajotelmaa. Katsotaan, minkä kokoisilla matriiseilla singulaariarvohajotelman hyödyntäminen on kannattavaa. Keskimääräiset laskenta-ajat `ode45`:lle on esitetty kuvassa 3.16 ja `ode15s`:lle kuvassa 3.17.



Kuva 3.16: Ratkaisijan ode45 keskimääräiset laskenta-ajat $p \times n$ -matriisille C, missä matriisin korkeus $p = 200$ ja leveyttä n kasvatetaan. Kullakin leveydellä n_i laskenta-aika on saatu 1000 simulaation keskiarvona, $n_i = 200, 250, \ldots, 1000$. Kuvaajaan on piirretty rasteilla $\times$ laskenta-ajat tapauksessa, jossa käytetään C:n konjugaattitranspoosia, eli kun $\widetilde{C}^* = C^*$ ja $B = -k\alpha^2 C^*$. Arvot on otettu kuvasta 3.5, joskin 200×200 -matriisin laskenta-aika on jätetty piirtämättä johtuen siitä, että laskentaan kului paljon enemmän aikaa kuin muun kokoisilla matriiseilla. Samaan kuvaajaan on myös piirretty timanteilla $\Diamond$ laskenta-ajat C:n pseudoinverssiä hyödynnettäessä, eli kun $\widetilde{C}^* = C^+$ ja $B = -k\alpha^2 C^+$. Kolmantena on vielä piirretty neliöillä $\square$ laskenta-ajat C:n skaalausmatriisia hyödynnettäessä, eli kun $\widetilde{C}^* = C^{\sigma_1^2}$ ja $B = -k\alpha^2 C^{\sigma_1^2}$. Neliömatriiseilla skaalausmatriisin käyttö on tehokkainta, kun taas leveillä matriiseilla on nopeinta käyttää konjugaattitranspoosia.

Verrataan ensin konjugaattitranspoosin C^* ja pseudoinverssin C^+ käytön tehokkuutta. Simulaatioiden perusteella näyttäisi siltä, että pseudoinverssin käyttö on kannattavampaa `ode45`-ratkaisijalle, kun $n \leq 2p$, ja `ode15s`-ratkaisijalle, kun $n \leq 3p$. Kuvien 3.16 ja 3.17 laskenta-ajoissa on kuitenkin laskettu vain differentiaa-



Kuva 3.17: Ratkaisijan ode15s keskimääräiset laskenta-ajat $p \times n$ -matriisille C, missä matriisin korkeus $p = 200$ ja leveyttä n kasvatetaan. Kullakin leveydellä n_i laskenta-aika on saatu 1000 simulaation keskiarvona, $n_i = 200, 250, \ldots, 1000$. Kuvaajaan on piirretty rasteilla $\times$ laskenta-ajat tapauksessa, jossa käytetään C:n konjugaattitranspoosia, eli kun $\widetilde{C}^* = C^*$ ja $B = -k\alpha^2 C^*$. Arvot on otettu kuvasta 3.6. Samaan kuvaajaan on myös piirretty timanteilla $\Diamond$ laskenta-ajat C:n pseudoinverssiä hyödynnettäessä, eli kun $\widetilde{C}^* = C^+$ ja $B = -k\alpha^2 C^{\sigma_1^2}$. Kolmantena on vielä piirretty neliöillä $\square$ laskenta-ajat C:n skaalausmatriisia hyödynnettäessä, eli kun $\widetilde{C}^* = C^{\sigma_1^2}$ ja $B = -k\alpha^2 C^{\sigma_1^2}$. Huomataan, että skaalausmatriisin käyttö on tehokkainta kaikissa tapauksissa.

liyhtälön ratkaisuun kuluva aika. Matriisin C singulaariarvohajotelman muodostamiseen menee myös oma aikansa, varsinkin suurilla matriiseilla, ja tätä ylimääräistä työtä ei ole otettu huomioon edellä mainituissa laskenta-ajoissa. Ajamalla ylimääräisiä simulaatioita suurilla matriiseilla huomataan, että pseudoinverssin käyttö on kannattavaa `ode45`:lle, kun $n \leq \frac{3}{2}p$, ja `ode15s`:lle, kun $n \leq 2p$. Joka tapauksessa `ode45`:n laskenta-ajat ovat lyhyemmät kuin `ode15s`:n laskenta-ajat silloin, kun käytetään C:n pseudoinverssiä. Leveillä matriiseilla pseudoinverssin käyttö ei ole tehokasta, mutta `ode45` on itsessään tehokas leveillä matriiseilla.

Pseudoinverssin C^+ ja skaalausmatriisin $C^{\sigma_1^2}$ vertailu on helppoa sekä `ode45`:llä että `ode15s`:llä. Kummallakin ratkaisijalla nimittäin skaalausmatriisin käyttö on kaikissa tilanteissa — niin neliömatriiseilla kuin leveillä matriiseilla — kannattavampaa kuin pseudoinverssin käyttö. Skaalausmatriisin hyödyntäminen on jokseenkin tehokkaampaa `ode45`:llä kuin `ode15s`:llä.

Viimeisenä verrataan konjugaattitranspoosin C^* ja skaalausmatriisin $C^{\sigma_1^2}$ käytön tehokkuutta. Ratkaisijalla `ode45` skaalausmatriisin käyttö on tehokkaampaa kuin konjugaattitranspoosin, kun $n \leq \frac{7}{2}p$. Skaalausmatriisin laskemisessa tarvitaan kuitenkin C:n singulaariarvohajotelmaa, aivan kuten pseudoinverssin C^+ laskemisessa. Kuvien 3.16 ja 3.17 laskenta-ajoissa on otettu huomioon vain differentiaaliyhtälön ratkaisuun kuluva aika eikä singulaariarvohajotelman muodostamiseen tarvittavaa ylimääräistä aikaa. Samoin kuin pseudoinverssiä käytettäessä, menetelmän valinnassa on syytä ottaa huomioon se, että singulaariarvohajotelman muodostaminen vaatii lisätyötä, varsinkin jos matriisit ovat dimensioiltaan suuria. Valitaan $\widetilde{C}^*$ skaalausmatriisiksi, kun $n \lesssim 2p$. Ratkaisijalla `ode15s` skaalausmatriisin hyödyntäminen on sekä neliömatriiseilla että leveillä matriiseilla laskennallisesti tehokkaampaa kuin konjugaattitranspoosin käyttö.

3.5.4 Yhteenveto parametrin $\widetilde{C}^*$ valinnasta

Kootaan yhteen tässä luvussa esitellyt tavat valita $\widetilde{C}^*$ matriisissa

$$A_c = \begin{bmatrix} A & B \\ C & O \end{bmatrix},$$

missä $A = -\alpha I_n$ ja $B = -k\alpha^2 \widetilde{C}^*$:

1. $\widetilde{C}^* = C^*$ (konjugaattitranspoosi). Laskennallisesti yksinkertaisin tapa muodostaa $\widetilde{C}^*$. Konjugaattitranspoosi on nopea muodostaa suurillakin matriiseilla, ja differentiaaliyhtälön ratkaisemisen kannalta menetelmä on nopea leveillä matriiseilla. Parametrin k valinnalla voidaan varmistaa, ettei systeemistä tule jäykkä tai että matriisin A_c ominaisarvot sijaitsevat halutussa sektorissa reaaliakseliin nähden, mutta yleensä molempia ominaisuuksia ei saada samanaikaisesti voimaan.

2. $\widetilde{C}^* = C^+$ (pseudoinverssi). Jos matriisilla C on singulaariarvohajotelma $C = U\Lambda V^*$ ja C:n singulaariarvoja ovat suuruusjärjestyksessä $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$, niin tulo $\sigma_i \tilde{\sigma}_i$ saa arvon 1, kun valitaan

$$\Lambda^+ = \begin{bmatrix} 1/\sigma_1 & 0 & \ldots & 0 \\ 0 & 1/\sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1/\sigma_p \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix},$$

ja asetetaan $C^+ = V\Lambda^+U^*$. Tällöin matriisin A_c ominaisarvot ovat muotoa

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i\frac{1}{\sigma_i}}\right) = \frac{\alpha}{2}\left(-1 \pm \sqrt{1-4k}\right), \quad i = 1, 2, \ldots, p.$$

Sopivilla parametrin k valinnoilla ominaisarvot saadaan haluttuun kulmaan reaaliakseliin nähden, eikä systeemi ole jäykkä. Pseudoinverssi nopeuttaa differentiaaliyhtälön ratkaisemista, kun C on neliömatriisi tai ainakin lähes neliömatriisi. Haittapuolena on, että singulaariarvohajotelman muodostaminen on laskennallisesti työlästä varsinkin suurikokoisille matriiseille.

3. $\widetilde{C}^* = C^{\sigma_1^2}$ (skaalausmatriisi). Kuten pseudoinverssin tapauksessa, skaalausmatriisin muodostaminen perustuu myös C:n singulaariarvohajotelmaan. Oletetaan jälleen, että $C = U\Lambda V^*$ ja että C:n singulaariarvot ovat suuruusjärjestyksessä $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$. Valitaan matriisi Λ^{a^2} seuraavalla tavalla:

$$\Lambda^{a^2} = \begin{bmatrix} a^2/\sigma_1 & 0 & \ldots & 0 \\ 0 & a^2/\sigma_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a^2/\sigma_p \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}.$$

Kun valitaan $a = \sigma_1$, saadaan muodostettua Λ:n skaalausmatriisi $\Lambda^{\sigma_1^2}$, ja sen jälkeen C:n skaalausmatriisi saadaan matriisitulosta $C^{\sigma_1^2} = V\Lambda^{\sigma_1^2}U^*$. Huomataan, että C:n pseudoinverssi on oikeastaan erikoistapaus skaalausmatriisista, ja se voidaan muodostaa valitsemalla $a = 1$. Valinnalla $a = \sigma_1$ matriisin A_c ominaisarvot ovat muotoa

$$\lambda_i = \frac{\alpha}{2}\left(-1 \pm \sqrt{1 - 4k\sigma_i\frac{\sigma_1^2}{\sigma_i}}\right) = \frac{\alpha}{2}\left(-1 \pm \sqrt{1-4k\sigma_1^2}\right), \quad i = 1, 2, \ldots, p.$$

Tällä lailla C:n pienten singulaariarvojen vaikutus saadaan poistettua, ja ominaisarvojen kannalta systeemi käyttäytyy ikään kuin C:n kaikki singulaariarvot olisivat σ_1. Parametrin k valinnalla A_c:n ominaisarvot saadaan haluttuun kulmaan reaaliakseliin nähden, ja systeemin jäykkyys saadaan myös estettyä. Skaalausmatriisi nopeuttaa differentiaaliyhtälön ratkaisua, kun C on neliömatriisi tai lähes sellainen, ja menetelmänä se toimii melko hyvin myös silloin, kun C on leveä matriisi. Kuten pseudoinverssin tapauksessa, skaalausmatriisin haittapuolena on, että singulaariarvohajotelman muodostaminen on laskennallisesti työlästä varsinkin suurikokoisille matriiseille.

Kootaan vielä ratkaisijoiden (`ode45` ja `ode15s`) ja menetelmien (konjugaattitranspoosi, pseudoinverssi ja skaalausmatriisi) tehokkuuksista yhteenveto eri matriisikokojen suhteen. Vertailukriteerinä on differentiaaliyhtälön ratkaisemiseen kuluva laskenta-aika.

- C on lähes neliömatriisi ($0 \leq n - p < 10$)
  - `ode45` on tehokkain ratkaisija, kun käytetään singulaariarvohajotelmaa, eli hyödynnetään C:n pseudoinverssiä tai skaalausmatriisia. Näistä kahdesta skaalausmatriisi on jokseenkin nopeampi vaihtoehto. Ilman singulaariarvohajotelmaa `ode45` on selvästi tehottomin.
  - `ode15s` on tehokas ratkaisija, ja vielä tehokkaampi, kun käytetään singulaariarvohajotelmaa. Skaalausmatriisilla differentiaaliyhtälön ratkaisu on hieman nopeampaa kuin pseudoinverssillä.
  - Tehokkain menetelmä on siis `ode45` singulaariarvohajotelmaa käyttävän skaalausmatriisin kanssa.
- C on leveä matriisi ($n - p \geq 10$)
  - `ode45` on tehokkaampi ratkaisija kuin `ode15s`. Mikäli $n \leq \frac{3}{2}p$, differentiaaliyhtälön ratkaisussa on tehokkaampaa käyttää pseudoinverssiä kuin konjugaattitranspoosia `ode45`:n kanssa. Skaalausmatriisi on aina nopeampi kuin pseudoinverssi differentiaaliyhtälön ratkaisussa, ja se on myös nopeampi kuin konjugaattitranspoosi, jos $n \lesssim 2p$. Muissa tapauksissa eli hyvin leveillä matriiseilla `ode45` toimii parhaiten, kun käytetään konjugaattitranspoosia.
  - Mikäli ratkaisijaa `ode15s` halutaan kuitenkin käyttää, pseudoinverssi on kannattavampi vaihtoehto kuin konjugaattitranspoosi `ode15s`:n kanssa, jos $n \leq 2p$. Jos taas $n > 2p$, konjugaattitranspoosi on näistä kahdesta parempi. Kuitenkin skaalausmatriisi on tehokkaampi kuin kumpikaan vaihtoehto `ode15s`:n kanssa.

Käytetään siis ratkaisijaa `ode45` aina differentiaaliyhtälön ratkaisemiseen. Jos $n \lesssim 2p$, käytetään skaalausmatriisia, mutta muuten käytetään konjugaattitranspoosia.

3.6 Lineaarisen yhtälöryhmän online-ratkaisu

Palataan luvussa 3.1 esitettyyn tutkimuskysymykseen, jossa lineaarista yhtälöryhmää $C_i\mathbf{x}_i = \mathbf{r}_i$ ratkaistaan toistuvasti siten, että muutokset peräkkäisten matriisien C_i ja vektorien $\mathbf{r}_i$ välillä ovat hyvin pienet. Ensin hahmotellaan DY-ratkaisijalle

periaate toistuvien ratkaisujen hakemiseksi, ja lopuksi esitetään yleinen ratkaisualgoritmi.

3.6.1 Online-ratkaisun hahmottelu

Tarkastellaan jälleen lineaarista yhtälöryhmää

$$C_1\mathbf{x}_1 = \mathbf{r}_1.$$

Yhtälöryhmän ratkaisu $\mathbf{x}_1$ voidaan hakea LU-hajotelman tai DY-ratkaisijan avulla. Oletetaan, että jälkimmäisellä menetelmällä ratkaisuksi on saatu $\mathbf{x}_1 = \mathbf{x}_1(t_1)$, missä t_1 on se ajanhetki, jolloin yhtälö (3.6) toteutuu, eli

$$\epsilon(t_1) = \frac{\|\mathbf{r}_1 - C_1\mathbf{x}_1(t_1)\|}{\|\mathbf{r}_1\|} < \epsilon_{\text{tol}}.$$

Seuraavaksi yhtälöryhmään kohdistetaan jokin seuraavista häiriöistä:

1. C_1:tä häiritään, $\mathbf{r}_1$ pysyy samana. Merkitään $C_2 = C_1 + \Delta C_1$ ja $\mathbf{r}_2 = \mathbf{r}_1$.
2. C_1 pysyy samana, $\mathbf{r}_1$:tä häiritään. Merkitään $C_2 = C_1$ ja $\mathbf{r}_2 = \mathbf{r}_1 + \Delta\mathbf{r}_1$.
3. Sekä C_1:tä että $\mathbf{r}_1$:tä häiritään. Merkitään $C_2 = C_1 + \Delta C_1$ ja $\mathbf{r}_2 = \mathbf{r}_1 + \Delta\mathbf{r}_1$.

Häiriön jälkeen ratkaistavana on uusi yhtälöryhmä

$$C_2\mathbf{x}_2 = \mathbf{r}_2.$$

LU-hajotelmalla yhtälöryhmä on ratkaistava kokonaan uudestaan — aikaisempaa ratkaisua $\mathbf{x}_1$ ei voida hyödyntää. Tarkastellaan seuraavaksi, miten differentiaaliyhtälösysteemi muuttuu häiriöistä. Merkitään differentiaaliyhtälöä

$$\mathbf{x}'_{c,2}(t) = A_{c,2}\mathbf{x}_{c,2}(t) + \mathbf{r}_{c,2},$$

missä

$$\mathbf{x}_{c,2}(t) = \begin{bmatrix}\mathbf{x}_2(t)\\ \xi_2(t)\end{bmatrix}, \quad A_{c,2} = \begin{bmatrix}-\alpha I_n & -k\alpha^2\widetilde{C}^*\\ C_2 & O\end{bmatrix} \quad \text{ja} \quad \mathbf{r}_{c,2} = \begin{bmatrix}\mathbf{0}\\ -\mathbf{r}_2\end{bmatrix}.$$

Tarkkaavainen lukija esittää nyt hyvän kysymyksen: mikä on matriisi $\widetilde{C}^*$? Ensimmäisellä "kierroksella" (ennen häiriöitä) valittiin $\widetilde{C}^* = C_1^*$ tai singulaariarvohajotelmaa hyödynnettäessä joko $\widetilde{C}^* = C_1^+$ tai $\widetilde{C}^* = C_1^{\sigma_1^2}$. Rajoitutaan tarkastelemaan tapausta, jossa singulaariarvohajotelmaa ei käytetty, eli ensimmäisellä kierroksella valittiin $\widetilde{C}^* = C_1^*$ (singulaariarvohajotelmaa käytettäessä seuraava tarkastelu on

täysin analoginen; konjugaattitranspoosi vain korvataan pseudoinverssillä tai skaalausmatriisilla). Toisella kierroksella voitaisiin päivittää parametreja α ja k sekä matriisia $\widetilde{C}^*$. Uudeksi matriisiksi saataisiin $\widetilde{C}^* = C_2^*$. Parametrien päivittäminen on kuitenkin laskennallisesti työlästä, ja systeemi halutaan muutenkin pitää mahdollisimman yksinkertaisena. Mikäli häiriö ΔC_1 oletetaan pieneksi (olettaen siis, että matriisia C_1 häirittiin), matriisi $A_{c,2}$ on stabiili, vaikkei parametreja päivitettäisikään. Valitaan siksi α ja k samalla tavalla kuin ensimmäisellä kierroksella, ja annetaan edelleen olla $\widetilde{C}^* = C_1^*$. Tällöin matriisissa $A_{c,2}$ on muuttunut matriisiin $A_{c,1}$ verrattuna ainoastaan vasemman alakulman lohko: $C_1 \to C_2$.

Nyt differentiaaliyhtälön ratkaisussa voidaan valita alkutilaksi (ajanhetkellä $t = t_1$)

$$\mathbf{x}_{c,2}(t_1) = \mathbf{x}_{c,1}(t_1),$$

eli ensimmäisellä kierroksella saatu ratkaisu. Pienillä häiriöillä uusi ratkaisu $\mathbf{x}_{c,2}(t_2)$ poikkeaa vain vähän vanhasta ratkaisusta $\mathbf{x}_{c,1}(t_1)$. Siksi uuden ratkaisun hakeminen on laskennallisesti nopeaa. Lineaarisella differentiaaliyhtälösysteemillä on se etu, että yhtälöryhmä voidaan edelleen ratkaista, vaikka kerroinmatriisia häirittäisiinkin. DY-ratkaisija on siis **robusti** häiriöiden suhteen. Robustisuudella tarkoitetaan yleisesti ottaen sitä, että ratkaisija toimii edelleen häiriöistä huolimatta, kunhan häiriöt pysyvät "tietyssä ympäristössä". Tässä tapauksessa kyseiseen ympäristöön kuuluvat sellaiset häiriöt, jotka eivät saata systeemiä epästabiiliksi.

Matriisiin $C := C_1$ kohdistetut häiriöt poikkeuttavat matriisin $A_{c,i}$ ominaisarvoja jonkin verran kompleksitasossa. Jos jokin ominaisarvoista sattuu siirtymään suljettuun oikeaan puolitasoon, eli systeemi muuttuu häiriöiden vaikutuksesta epästabiiliksi, systeemi saadaan uudelleen stabiiliksi päivittämällä matriisia $\widetilde{C}^*$. Päivitys tapahtuu siten, että $\widetilde{C}^*$ valitaan häirityn C:n konjugaattitranspoosiksi (singulaariarvohajotelmaa hyödynnettäessä pseudoinverssiksi tai skaalausmatriisiksi): $\widetilde{C}^* = C_k^*$ (tai $\widetilde{C}^* = C_k^+$ tai $\widetilde{C}^* = C_k^{\sigma_1^2}$), missä C_k on $(k-1)$ kertaa häiritty matriisi C. Riittävän pienillä häiriöillä ominaisarvot pysyvät avoimessa vasemmassa puolitasossa ja systeemi pysyy stabiilina, mutta jos häiriöitä on paljon, voi olla järkevää aika ajoin (esimerkiksi 100 häiriön välein, tai jos laskenta alkaa merkittävästi hidastua) päivittää matriisia $\widetilde{C}^*$ häirityn C:n avulla. Näin varmistutaan, etteivät ominaisarvot karkaa stabiililta puolelta liian lähelle imaginaariakselia. Toisaalta, jos häiritty C

on muotoa

$$\begin{aligned} C_{k+1} &= C_k + \Delta C_k \\ &= (C_{k-1} + \Delta C_{k-1}) + \Delta C_k \\ &= (C_{k-2} + \Delta C_{k-2}) + \Delta C_{k-1} + \Delta C_k \\ &\vdots \\ &= (C_1 + \Delta C_1) + \Delta C_2 + \ldots + \Delta C_k \\ &= C_1 + (\Delta C_1 + \Delta C_2 + \ldots + \Delta C_k), \end{aligned}$$

missä häiriöt ΔC_i oletetaan riippumattomiksi ja normaalijakautuneiksi, niin häiriöiden summa $\Delta C_1 + \ldots + \Delta C_k$ on myös normaalijakautunut [19, s. 59], ja monesti häiriöt ainakin jossain määrin kumoavat toisensa. Huomaa, että jos matriisia C ei häiritä ollenkaan, vaan kaikki häiriöt kohdistuvat vektoriin $\mathbf{r} := \mathbf{r}_1$, systeemi pysyy koko ajan stabiilina, ja ainoa muuttuva termi systeemissä on $\mathbf{r}_{c,i}$. Tällöin matriisia $A_c := A_{c,1}$ ei tarvitse missään vaiheessa päivittää.

Ensimmäisen häiriön jälkeen systeemiä voidaan häiritä uudelleen. Merkitään jälleen $C_3 = C_2 + \Delta C_2$ ja $\mathbf{r}_3 = \mathbf{r}_2 + \Delta\mathbf{r}_2$. Jos C_2:ta ei häiritäkään, niin $\Delta C_2 = O$, ja jos $\mathbf{r}_2$:ta ei häiritä, niin $\Delta\mathbf{r}_2 = \mathbf{0}$. Uudeksi differentiaaliyhtälöksi saadaan

$$\mathbf{x}'_{c,3}(t) = A_{c,3}\mathbf{x}_{c,3}(t) + \mathbf{r}_{c,3},$$

missä

$$\mathbf{x}_{c,3}(t) = \begin{bmatrix} \mathbf{x}_3(t) \\ \xi_3(t) \end{bmatrix}, \quad A_{c,3} = \begin{bmatrix} -\alpha I_n & -k\alpha^2\widetilde{C}^* \\ C_3 & O \end{bmatrix} \quad \text{ja} \quad \mathbf{r}_{c,3} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{r}_3 \end{bmatrix}.$$

Matriisissa $A_{c,3}$ vain vasemman alakulman lohkoa eli matriisia C_3 on häiritty. Oikean yläkulman lohkossa valitaan edelleen $\widetilde{C}^* = C_1^*$ ($\widetilde{C}^* = C_1^+$ tai $\widetilde{C}^* = C_1^{\sigma_1^2}$, jos käytetään singulaariarvohajotelmaa). Häiriö ei siis vaikuta $\widetilde{C}^*$:n valintaan. Jos ensimmäisen häiriön jälkeen systeemille saatiin ajanhetkellä t_2 ratkaisu $\mathbf{x}_{c,2} = \mathbf{x}_{c,2}(t_2)$, niin asetetaan häiritylle systeemille alkutilaksi

$$\mathbf{x}_{c,3}(t_2) = \mathbf{x}_{c,2}(t_2).$$

Differentiaaliyhtälön ratkaisussa voidaan siis jälleen hyödyntää edellisen ratkaisun lopputilaa. Sen sijaan LU-hajotelmaa käytettäessä ratkaisu on haettava kokonaan uudelleen. Sama periaate toistuu seuraavillakin kierroksilla, kun systeemiin kohdistetaan jälleen häiriöitä. Siksi voidaan muodostaa yleinen algoritmi häirityn differentiaaliyhtälösysteemin ratkaisulle.

3.6.2 Online-algoritmi

1. Asetetaan $C_1 = C$, $\widetilde{C}^* = C^*$ (tai vastaavasti $\widetilde{C}^* = C^+$ tai $\widetilde{C}^* = C^{\sigma_1^2}$),

$$A_{c,1} = A_c = \begin{bmatrix} -\alpha I_n & -k\alpha^2 \widetilde{C}^* \\ C & O \end{bmatrix} \quad \text{ja} \quad \mathbf{r}_{c,1} = \mathbf{r}_c = \begin{bmatrix} \mathbf{0} \\ -\mathbf{r} \end{bmatrix}.$$

   Lisäksi valitaan ajanhetkellä $t_0 = 0$ ensimmäistä alkutilaa varten jokin vektori $\mathbf{x}_{c,0}(t_0) = \mathbf{x}_{c,0}$.

2. Aloitetaan i:s kierros muodostamalla differentiaaliyhtälö

$$\mathbf{x}'_{c,i}(t) = A_{c,i}\mathbf{x}_{c,i}(t) + \mathbf{r}_{c,i}, \tag{3.41a}$$

   missä

$$\mathbf{x}_{c,i}(t) = \begin{bmatrix} \mathbf{x}_i(t) \\ \xi_i(t) \end{bmatrix}, \quad A_{c,i} = \begin{bmatrix} -\alpha I_n & -k\alpha^2 \widetilde{C}^* \\ C_i & O \end{bmatrix} \quad \text{ja} \quad \mathbf{r}_{c,i} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{r}_i \end{bmatrix}, \tag{3.41b}$$

   ja $i = 1, 2, \ldots, N$. Valitaan alkutilaksi

$$\mathbf{x}_{c,i}(t_{i-1}) = \mathbf{x}_{c,i-1}(t_{i-1}).$$

3. Jos systeemi osoittautuu simulaatiossa epästabiiliksi (ratkaisun suhteellinen virhe kasvaa ylärajaa M suuremmaksi), valitaan $\widetilde{C}^* = C_i^*$ ($\widetilde{C}^* = C_i^+$ tai $\widetilde{C}^* = C_i^{\sigma_1^2}$), päivitetään parametrit α ja k ja sijoitetaan päivitetyt $\widetilde{C}^*$, α ja k matriisiin $A_{c,i}$. Muussa tapauksessa siirrytään suoraan vaiheeseen 4.

4. Ratkaistaan differentiaaliyhtälö (3.41) käyttämällä jotain DY-ratkaisijaa. Kun jollain ajanhetkellä t_i on voimassa

$$\epsilon(t_i) = \frac{\|\mathbf{r}_i - C_i\mathbf{x}_i(t_i)\|}{\|\mathbf{r}_i\|} < \epsilon_{\text{tol}},$$

   asetetaan numeeriseksi ratkaisuksi

$$\mathbf{x}_{c,i} = \mathbf{x}_{c,i}(t_i).$$

   Saatu lopputila on myös seuraavan kierroksen alkutila.

5. Häiritään matriisia C_i ja/tai vektoria $\mathbf{r}_i$:

$$C_{i+1} = C_i + \Delta C_i, \qquad \mathbf{r}_{i+1} = \mathbf{r}_i + \Delta \mathbf{r}_i.$$

Häiriöissä voi olla $\Delta C_i = O$ tai $\Delta \mathbf{r}_i = \mathbf{0}$. Muita systeemin osia ei muuteta.

6. Jos $i < N$, jatketaan seuraavalle kierrokselle $(i + 1)$ siirtymällä kohtaan 2. Tapauksessa $i = N$ algoritmi päättyy.

Edellä olevassa algoritmissa kuvattua menetelmää sanotaan lineaarisen yhtälöryhmän **online**-ratkaisuksi. Huomaa, että matriisi $\widetilde{C}^*$ voidaan valita (vähintään) kolmella eri tavalla, joko C:n konjugaattitranspoosiksi C^*, pseudoinverssiksi C^+ tai skaalausmatriisiksi $C^{\sigma_1^2}$. Kullakin valintatavalla on erilaiset parametrit, joten useaa eri valintatapaa ei kannata käyttää saman algoritmiajon aikana. Toisin sanottuna, valitaan yksi näistä kolmesta matriisityypistä, ja sen jälkeen käytetään johdonmukaisesti vain sitä.

Lopuksi mainittakoon, että lineaarisen yhtälöryhmän robusti online-ratkaisu lineaarisen differentiaaliyhtälösysteemin avulla voidaan yleistää ääretönulotteisille systeemeille [23].

4. RATKAISIJOIDEN VERTAILUA

Verrataan DY-ratkaisijan ja LU-hajotelman ratkaisuaikoja keskenään sivulla 68 esitetyn online-algoritmin avulla. Vaiheessa 4 haetaan differentiaaliyhtälön (3.41) ratkaisu jollain DY-ratkaisijalla, esimerkiksi `ode45`:llä, ja otetaan aikaa, kuinka kauan ratkaisun hakeminen kestää. Vertailun vuoksi ratkaistaan samassa vaiheessa lineaarinen yhtälöryhmä $C_i\mathbf{x}_i = \mathbf{r}_i$ LU-hajotelman avulla, ja otetaan jälleen aikaa, kuinka kauan yhtälöryhmän ratkaiseminen kestää. Pyritään selvittämään, millaisissa tilanteissa DY-ratkaisija on tehokkaampi kuin LU-hajotelma, ja sama toisin päin.

4.1 Periaate ratkaisijoiden vertailulle

Kootaan ensin yhteen ratkaisijoissa käytettävät parametrit. Sen jälkeen muotoillaan simulaatioissa käytettävät häiriöt, ja määritetään myös, missä tapauksissa parametrit on syytä päivittää ratkaisijoiden tehokkuuden parantamiseksi. Lopuksi esitetään vertailualgoritmi LU-hajotelmalle ja DY-ratkaisijalle.

4.1.1 Parametrien kertaus

Ennen ratkaisijoiden vertailua kootaan yhteen, miten lineaarisessa differentiaaliyhtälösysteemissä käytettävät parametrit kannattaa valita, jotta DY-ratkaisijalla laskenta olisi mahdollisimman tehokasta ja laskenta-ajat mahdollisimman lyhyet. Matriisi C ja vektori $\mathbf{r}$ oletetaan annetuiksi. Simulaatioissa ne valitaan satunnaisesti siten, että C:n ja $\mathbf{r}$:n alkiot noudattavat normaalijakaumaa N(0, 1). Matriisin C koko $p \times n$ voi vaihdella siten, että C on joko neliömatriisi tai leveä matriisi, $p \leq n$. Käytetään simulaatioissa suurikokoisia matriiseja ($n \geq p \geq 100$), koska liian pienillä matriiseilla kummankin ratkaisijan laskenta-ajat ovat niin lyhyet, ettei aikoja käytännössä pysty vertailemaan. Valitaan alkutila $\mathbf{x}_{c,0}(t_0) = \mathbf{x}_{c,0}$ myös satunnaisesti siten, että alkutilan kukin komponentti noudattaa N(0, 1) -normaalijakaumaa. Systeemissä (3.41) valitaan parametri α kaavan (3.27) mukaisesti, `ode45`-ratkaisijalle $\alpha = 0{,}01$ ja `ode15s`:lle $\alpha = 10$. Parametrin k valinta riippuu siitä, miten $\widetilde{C}^*$ valitaan. Luvussa 3.5 esiteltiin kolme erilaista tapaa, miten $\widetilde{C}^*$ voidaan esimerkiksi valita: matriisin C konjugaattitranspoosiksi, pseudoinverssiksi tai skaalausmatriisiksi. Jos $\widetilde{C}^* = C^*$,

niin parametri k valitaan DY-ratkaisijan perusteella (kaavat (3.25) ja (3.26)):

$$k_{\text{ode45}} = \frac{1}{8}\left(\frac{1}{\sigma_{p-1}^2} + \frac{1}{\sigma_p^2}\right), \quad \text{ja} \quad k_{\text{ode15s}} = \frac{1}{4\sigma_1^2},$$

missä σ_i:t ovat C:n singulaariarvoja suuruusjärjestyksessä, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$. Jos taas $\widetilde{C}^* = C^+$, niin valitaan kaavan (3.31) mukaan $k = 0{,}325$. Jos valitaankin $\widetilde{C}^* = C^{\sigma_1^2}$, niin asetetaan k kaavan (3.39) mukaisesti DY-ratkaisijan perusteella:

$$k_{\text{ode45}} = \frac{0{,}475}{\sigma_1^2} \quad \text{ja} \quad k_{\text{ode15s}} = \frac{0{,}375}{\sigma_1^2}.$$

Simulaatioissa pyritään yleensä noudattamaan luvussa 3.5 havaittua periaatetta: neliömatriiseilla tai lähes neliömatriiseilla ($n \approx p$) valitaan $\widetilde{C}^* = C^{\sigma_1^2}$ (voidaan myös kokeilla vaihtoehtoa $\widetilde{C}^* = C^+$), leveillä matriiseilla ($n \gg p$) valitaan $\widetilde{C}^* = C^*$. Toleranssiksi ϵ_{tol} valitaan

$$\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}.$$

Simulaatioissa häiritään yleensä vain matriisia C. Toisin sanottuna, $C_{i+1} = C_i + \Delta C_i$ ja $\mathbf{r}_{i+1} = \mathbf{r}_i = \mathbf{r}$. Vektorin $\mathbf{r}$ häiritseminen on käytännössä triviaalia, koska LU-hajotelmaa käytettäessä $\mathbf{r}$:n häiritseminen ei muuta C:n LU-hajotelmaa ja DY-ratkaisijaa käytettäessä $\mathbf{r}$:n häiritseminen ei muuta matriisia A_c. Pyritään pitämään häiriöt pieninä. Häiriömatriisin ΔC_i alkiot on siis valittava itseisarvoltaan huomattavasti pienemmiksi kuin matriisin C_i alkiot.

4.1.2 Häiriöiden muoto simulaatioissa

Pohditaan seuraavaksi, millaiset häiriöt ΔC ja $\Delta\mathbf{r}$ olisivat sopivia. Oletetaan, että ajanhetkellä t ratkaisun (suhteelliselle) virheelle on voimassa

$$\frac{\|C\mathbf{x}(t) - \mathbf{r}\|}{\|\mathbf{r}\|} < \epsilon_{\text{tol}}.$$

Merkitään vektoria $\mathbf{x}(t)$ lyhyesti vain $\mathbf{x}$:llä. Kun ajanhetkellä t systeemiin kohdistetaan häiriö siten, että vaihdetaan $C \to C + \Delta C$, ratkaisun virhettä häiritylle

systeemille voidaan arvioida sillä ajanhetkellä:

$$\begin{aligned}\frac{\|(C+\Delta C)\mathbf{x}(t)-\mathbf{r}\|}{\|\mathbf{r}\|} &= \frac{\|C\mathbf{x}-\mathbf{r}+\Delta C\mathbf{x}\|}{\|\mathbf{r}\|}\\ &\leq \frac{\|C\mathbf{x}-\mathbf{r}\|}{\|\mathbf{r}\|}+\frac{\|\Delta C\mathbf{x}\|}{\|\mathbf{r}\|}\\ &\leq \frac{\|C\mathbf{x}-\mathbf{r}\|}{\|\mathbf{r}\|}+\frac{\|\Delta C\|\,\|\mathbf{x}\|}{\|\mathbf{r}\|}\\ &< \epsilon_{\text{tol}}+\frac{\|\Delta C\|\,\|\mathbf{x}\|}{\|\mathbf{r}\|}.\end{aligned}$$

Simulaatioissa tarkastellaan sellaisia tilanteita, joissa ratkaisun suhteellinen virhe ei merkittävästi muutu. Oletetaan häiriöt sen verran pieniksi, että suhteellinen virhe häiriön jälkeen on esimerkiksi välillä $(\epsilon_{\text{tol}},\, 2\epsilon_{\text{tol}})$.

4.1.3 Parametrien päivityksestä

Palautetaan vielä mieleen, että jos matriisia $C = C_1$ häiritään i kertaa, niin matriisi $A_{c,(i+1)}$ on muotoa

$$A_{c,(i+1)} = \begin{bmatrix} -\alpha I_n & -k\alpha^2\widetilde{C}^* \\ C_i + \Delta C_i & O \end{bmatrix},$$

missä $\widetilde{C}^*$ on joko C_j^*, C_j^+ tai $C_j^{\sigma_1^2}$. Tässä $j \leq i$ on se indeksi, jolla parametri $\widetilde{C}^*$ valittiin edellisen kerran (alussa $j = 1$). Tavallisesti parametreja α, k ja $\widetilde{C}^*$ ei päivitetä, vaikka C:tä häiritäänkin, koska parametrien päivittäminen vaatii laskennallisesti lisätyötä. Tarkastellaan tilanteita, joissa parametreja kannattaa tai on pakko päivittää. Mikäli C:n häiriöt saattavat systeemin epästabiiliksi, eli jokin A_c:n ominaisarvoista siirtyy kompleksitasossa suljettuun oikeaan puolitasoon, parametrit on päivitettävä, jotta differentiaaliyhtälö voidaan jälleen ratkaista. Matriisin $\widetilde{C}^*$ päivitys i:nnen häiriön jälkeen tapahtuu seuraavalla tavalla:

- Jos $\widetilde{C}^* = C_j^*$, vaihdetaan $C_j^* \to (C_i + \Delta C_i)^*$.
- Jos $\widetilde{C}^* = C_j^+$, vaihdetaan $C_j^+ \to (C_i + \Delta C_i)^+$.
- Jos $\widetilde{C}^* = C_j^{\sigma_1^2}$, vaihdetaan $C_j^{\sigma_1^2} \to (C_i + \Delta C_i)^{\sigma_1^2}$.

Parametria α ei käyttämillämme valinnoilla tarvitse päivittää ollenkaan, koska α valittiin vakioksi. Parametri k päivitetään sen mukaan, valittiinko $\widetilde{C}^*$ häirityn matriisin $C_i + \Delta C_i$ konjugaattitranspoosiksi, pseudoinverssiksi vai skaalausmatriisiksi. Huomaa, että k:n lausekkeessa usein esiintyvät singulaariarvot σ_l otetaan nyt häirityn matriisin $C_i + \Delta C_i$ singulaariarvoista.

Toinen tilanne, kun matriisia $\widetilde{C}^*$ tarvitsee päivittää, on silloin, kun A_c:n ominaisarvot ovat häiriöiden vaikutuksesta siirtyneet liian kauas alkuperäisestä sijainnistaan kompleksitasossa. Tällöin alkuperäiselle systeemille säädetyt parametrit eivät enää toimi yhtä tehokkaasti häirityille systeemeille kuin alkuperäiselle systeemille. Osa ominaisarvoista saattaa myös siirtyä hyvin lähelle imaginaariakselia, jolloin systeemi on lähes epästabiili, mistä johtuen tasapainotila saavutetaan äärimmäisen hitaasti. Sekä parametrisäätöjen heikkeneminen että siirtyminen kohti epästabiilia systeemiä ilmenevät DY-ratkaisijassa samalla tavalla: laskenta-ajat alkavat hidastua. Koska ominaisarvojen sijainnin ja muutoksen tarkkailu on varsinkin suurilla matriiseilla laskennallisesti työlästä, tarvitaan jokin toinen kriteeri, josta tiedetään parametrien tarvitsevan päivitystä. Valitaan täksi kriteeriksi laskenta-ajat. Mikäli differentiaaliyhtälön ratkaisemiseen kuluva aika alkaa merkittävästi pidentyä, päivitetään parametrit α, k ja $\widetilde{C}^*$ laskennan nopeuttamiseksi. Päivitykset tehdään edellisessä kappaleessa mainituilla tavoilla.

Paitsi että parametrien päivittäminen vaatii aina laskennallista lisätyötä, se myös suurentaa ratkaisun virhettä. Jos matriisissa A_c muutetaan kahta eri lohkoa, $B = -k\alpha^2\widetilde{C}^*$ ja C, myös differentiaaliyhtälön ratkaisu $\mathbf{x}_c(t) = \begin{bmatrix}\mathbf{x}(t)\\ \xi(t)\end{bmatrix}$ muuttuu, ja lineaarisen yhtälöryhmän ratkaisun virhe on suurempi kuin siinä tapauksessa, että pelkästään matriisia C häiritään. Seurauksena ratkaisun hakeminen parametripäivityksen jälkeen on laskenta-aikojen kannalta hidasta. Pyritään siksi löytämään sopiva päivitystiheys siten, että parametrit ja ominaisarvojen sijainti eivät pääse "huononemaan" liian paljon, mutta kuitenkin siten, että päivitystä ei suoriteta liian usein. Nyrkkisääntönä voidaan pitää esimerkiksi, että parametrit kannattaa päivittää noin sadan häiriön välein. Kokeilemalla huomataan, että laskenta-aikojen kasvaminen suurin piirtein 3,0-kertaiseksi on sopiva kriteeri. Jos siis edellisiin ratkaisuihin verrattuna laskenta-aika kasvaa 3,0-kertaiseksi, asetetaan parametrit päivitettäväksi. Noin kymmenen edellistä laskenta-aikaa riittää hyvin vertailukohteeksi; tällaisella otoksella pystyy jo havaitsemaan sen, jos laskenta-ajat alkavat kasvaa. Näillä valinnoilla päivitystiheys saadaan sopivaksi, ja parametreja päivitetään noin sadan häiriön välein. Huomaa, että jos matriisia C ei häiritä ollenkaan ($\Delta C = O$), parametreja ei tarvitse missään vaiheessa päivittää.

4.1.4 Vertailualgoritmi

Kootaan vielä yhteenveto menetelmästä, jolla ratkaisijoita verrataan. Tämä vertailu perustuu sivulla 68 esitettyyn online-algoritmiin sekä tässä luvussa mainittuihin lisäsäätöihin.

1. Asetetaan $C_1 = C = \mathrm{randn}(p, n)$ ja $\mathbf{r}_1 = \mathbf{r} = \mathrm{randn}(p, 1)$. Jos $n/p < 2$, käytetään C:n singulaariarvohajotelmaa. Jos $n/p > 2$, käytetään C:n konjugaat-

titranspoosia. Rajalla $n/p = 2$ käytetään jompaa kumpaa. Asetetaan $t_0 = 0$, ja valitaan alkutilaksi $\mathbf{x}_{c,0}(t_0) = \text{randn}(n + p, 1)$ sekä ratkaisun suhteellisen virheen toleranssiksi $\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}$.

2. Aloitetaan i:s kierros. Jos kierroksia on N kappaletta, niin $i = 1, 2, \ldots, N$. Olkoon $\alpha = 0{,}01$ ja $A_i = -\alpha I_n$. Merkitään C_i:n singulaariarvoja suuruusjärjestyksessä $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p > 0$. Jos halutaan käyttää singulaariarvohajotelmaa, muodostetaan C_i:n singulaariarvohajotelma siten, että $C_i = U_i \Lambda_i V_i^*$. Asetetaan sitten C_i:n skaalausmatriisin avulla $\widetilde{C}^* = C_i^{\sigma_1^2} = V_i \Lambda_i^{\sigma_1^2} U_i^*$ (vaihtoehtoisesti tapauksessa $n \approx p$ voidaan käyttää C_i:n pseudoinverssiä: $\widetilde{C}^* = C_i^+ = V_i \Lambda_i^+ U_i^*$). Skaalausmatriisia käytettäessä valitaan $k = \frac{0{,}475}{\sigma_1^2}$, ja pseudoinverssiä käytettäessä valitaan $k = 0{,}325$. Jos singulaariarvohajotelmaa ei haluta hyödyntää, asetetaan C_i:n konjugaattitranspoosin avulla $\widetilde{C}^* = C_1^*$ ja valitaan $k = \frac{1}{8}\left(\frac{1}{\sigma_{p-1}^2} + \frac{1}{\sigma_p^2}\right)$.

   - Huomaa, että parametrit α, k ja $\widetilde{C}^*$ asetetaan vain, jos niitä ei ole vielä asetettu tai jos parametrit tarvitsee päivittää. Jälkimmäinen tapaus esiintyy useimmiten silloin, jos laskenta-ajat ovat merkittävästi hidastuneet. Sovitaan kriteeriksi, että jos laskenta-ajat ovat viimeisen 10 kierroksen aikana kasvaneet 3,0-kertaiseksi, päivitetään parametrit α, k ja $\widetilde{C}^*$.

   Asetetaan $B_i = -k\alpha^2 \widetilde{C}^*$ ja $D_i = O$. Muodostetaan matriisi

$$A_{c,i} = \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix} = \begin{bmatrix} -\alpha I_n & -k\alpha^2 \widetilde{C}^* \\ C_i & O \end{bmatrix}$$

   ja vektori

$$\mathbf{r}_{c,i} = \begin{bmatrix} \mathbf{0} \\ -\mathbf{r}_i \end{bmatrix}.$$

   Asetetaan alkutilaksi

$$\mathbf{x}_{c,i}(t_{i-1}) = \mathbf{x}_{c,i-1}(t_{i-1}).$$

   Differentiaaliyhtälössä ratkaistavana on vektori

$$\mathbf{x}_{c,i}(t) = \begin{bmatrix} \mathbf{x}_i(t) \\ \xi_i(t) \end{bmatrix}.$$

3. Muodostetaan differentiaaliyhtälö

$$\mathbf{x}'_{c,i}(t) = A_{c,i}\mathbf{x}_{c,i}(t) + \mathbf{r}_{c,i}. \tag{4.1}$$

   Yritetään ratkaista differentiaaliyhtälö `ode45`:llä.

- Olkoon M jokin valittavissa oleva suuri luku, esimerkiksi $M = 1{,}0 \cdot 10^6$. Jos jollain ajanhetkellä t_M on voimassa

$$\epsilon(t_M) = \frac{\|\mathbf{r}_i - C_i\mathbf{x}_i(t_M)\|}{\|\mathbf{r}_i\|} > M,$$

niin todetaan systeemi epästabiiliksi. Päivitetään parametrit α, k ja $\widetilde{C}^*$, ja aloitetaan i:s kierros alusta palaaamalla kohtaan 2.

Mikäli systeemi on stabiili, jollain ajanhetkellä t_i on voimassa

$$\epsilon(t_i) = \frac{\|\mathbf{r}_i - C_i\mathbf{x}_i(t_i)\|}{\|\mathbf{r}_i\|} < \epsilon_{\text{tol}}.$$

Asetetaan tällöin differentiaaliyhtälön numeeriseksi ratkaisuksi

$$\mathbf{x}_{c,i} = \mathbf{x}_{c,i}(t_i).$$

Saatu lopputila on myös seuraavan kierroksen alkutila. Lineaarisen yhtälöryhmän $C_i\mathbf{x}_i = \mathbf{r}_i$ numeerinen ratkaisu on

$$\mathbf{x}_i = \mathbf{x}_i(t_i).$$

Mitataan differentiaaliyhtälösysteemin (4.1) ratkaisemiseen kulunut laskenta-aika T_i^{DY}.

4. Ratkaistaan lineaarinen yhtälöryhmä

$$C_i\mathbf{x}_i = \mathbf{r}_i \tag{4.2}$$

LU-hajotelman avulla. Mitataan yhtälöryhmän ratkaisemiseen kulunut laskenta-aika T_i^{LU}.

- Huomaa, että DY-ratkaisijalla ja LU-hajotelmalla saadut ratkaisut eivät yleensä ole samat, sillä yhtälöllä $C_i\mathbf{x}_i = \mathbf{r}_i$ ei ole yksikäsitteistä ratkaisua, kun C on leveä matriisi. Sen sijaan jos C on neliömatriisi, ratkaisut ovat samat, jos numeerista laskentatarkkuutta ei oteta huomioon.

5. Häiritään matriisia C_i siten, että häiriön jälkeen ratkaisun virhe $\epsilon(t_i)$ on välillä

$$\epsilon_{\text{tol}} < \epsilon(t_i) < 2\epsilon_{\text{tol}}.$$

6. Tavallisesti vektoria $\mathbf{r}_i$ ei häiritä, ellei toisin mainita. Asetetaan siis $\Delta\mathbf{r}_i = \mathbf{0}$ ja $\mathbf{r}_{i+1} = \mathbf{r}_i$, ja siirrytään kohtaan 7. Jos vektoria $\mathbf{r}_i$ kuitenkin halutaan häiritä,

toteutetaan häiriöt siten, että niiden jälkeen (kun C_i:tä ja/tai $\mathbf{r}_i$:tä on häiritty) ratkaisun virhe $\epsilon(t_i)$ on välillä

$$\epsilon_{\text{tol}} < \epsilon(t_i) < 2\epsilon_{\text{tol}}.$$

7. Jos $i < N$, jatketaan seuraavalle kierrokselle $(i + 1)$ siirtymällä kohtaan 2. Tapauksessa $i = N$ siirrytään kohtaan 8.

8. Lasketaan kokonaislaskenta-ajat DY-ratkaisijalle ja LU-hajotelmalle:

$$T_{\text{kok}}^{\text{DY}} = \sum_{i=1}^{N} T_i^{\text{DY}} \qquad \text{ja} \qquad T_{\text{kok}}^{\text{LU}} = \sum_{i=1}^{N} T_i^{\text{LU}}. \tag{4.3}$$

Ratkaisijoiden vertailu toteutetaan hyvin yksinkertaisella tavalla: se ratkaisija on tehokkaampi, jolla on pienempi kokonaislaskenta-aika. Lopuksi piirretään kaksi kuvaajaa. Ensimmäiseen kuvaajaan piirretään yhtenäisellä viivalla ratkaisun suhteellinen virhe $\epsilon(t)$ systeemin ajan t funktiona. Virhefunktioon $\epsilon(t)$ on lisätty timantti $\Diamond$ siihen kohtaan, jossa systeemiä (matriisia C_i ja/tai vektoria $\mathbf{r}_i$) on häiritty, ja ylöspäin osoittava kolmio $\triangle$ siihen kohtaan, jossa systeemin parametrit α, k ja $\widetilde{C}^*$ on päivitetty. Samaan kuvaajaan on piirretty katkoviivalla toleranssiraja ϵ_{tol}. Toiseen kuvaajaan piirretään DY-ratkaisijan ja LU-hajotelman kokonaislaskenta-ajat $\sum_{i=1}^{k} T_i^{\text{DY}}$ ja $\sum_{i=1}^{k} T_i^{\text{LU}}$ kierroksen k funktiona. DY-ratkaisijan kokonaislaskenta-aika on piirretty yhtenäisellä viivalla, LU-hajotelman katkoviivalla. Kuvaajien piirtämisen jälkeen algoritmi päättyy.

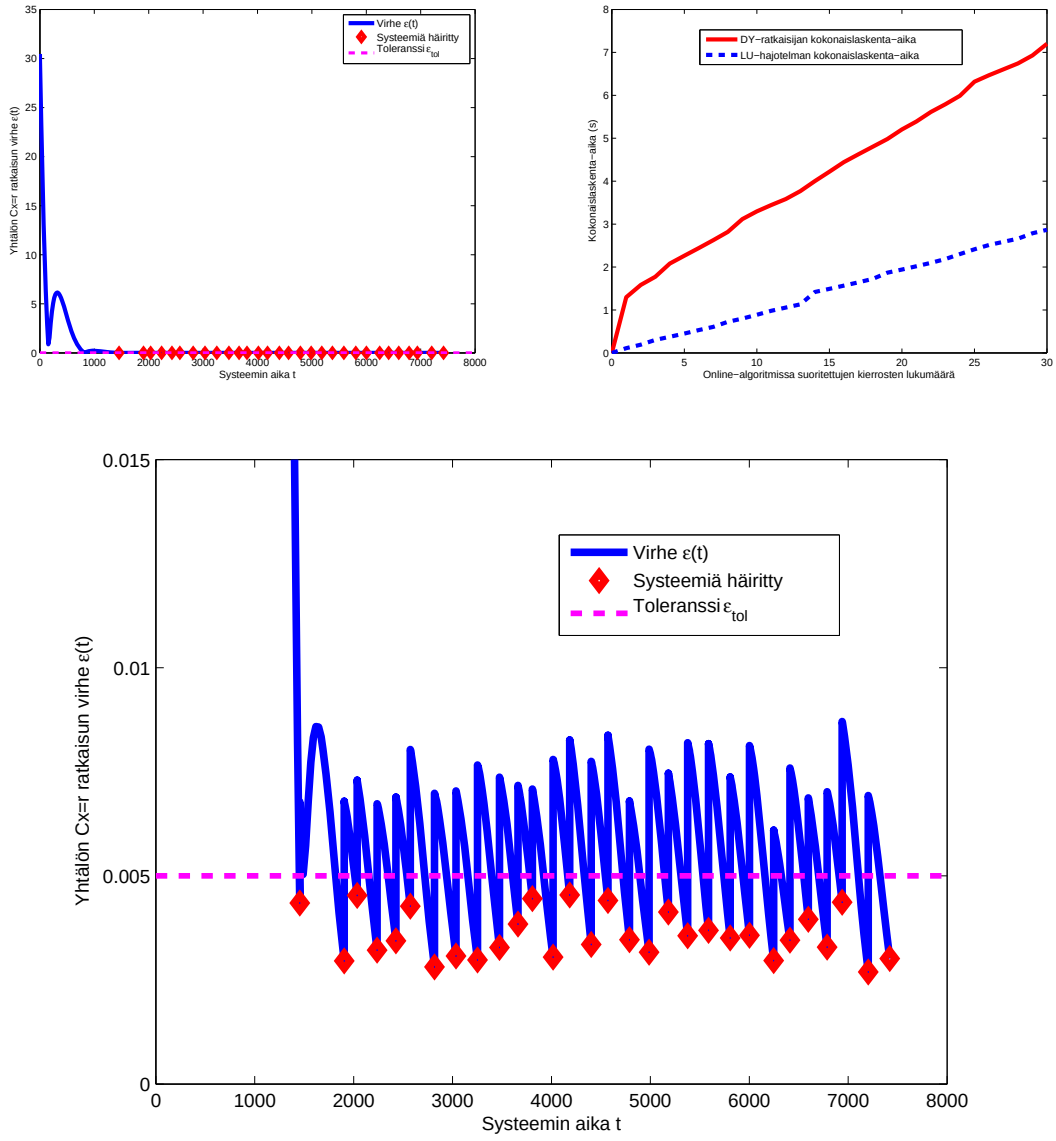
4.2 Simulaatioita

Sovelletaan seuraavaksi ratkaisijoiden vertailualgoritmia useassa eri esimerkkitapauksessa. Käytetään näissä esimerkeissä eri kokoisia matriiseja. Pyritään selvittämään, millaisissa tilanteissa LU-hajotelma on tehokkaampi kuin DY-ratkaisija, ja toisinpäin. Simulaatioissa käytetään MATLAB:ia.

4.2.1 Neliömatriisi

Esimerkki 4.2.1. Olkoon C 1000×1000 -matriisi ja $\mathbf{r}$ 1000×1 -vektori, ja oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Verrataan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja 30 kierroksen ajan, kun matriisia C häiritään. Haetaan siis yhtälön $C\mathbf{x} = \mathbf{r}$ ratkaisu $\mathbf{x}$ kummallakin ratkaisijalla, häiritään C:tä, ratkaistaan yhtälö jälleen, häiritään C:tä, ja näin jatketaan, kunnes

C:tä on häiritty 30 kertaa. Differentiaaliyhtälön ratkaisemisessa käytetään skaalausmatriisimenetelmää, eli valitaan $\widetilde{C}^* = C\sigma_1^2$. Ratkaisun suhteellinen virhe $\epsilon(t)$ DY-ratkaisijalle sekä menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.1.



Kuva 4.1: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1000×1000 -matriisi. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, alhaalla on sama kuvaaja suurennettuna ja eri skaalauksella, ja yläoikealla on ratkaisijoiden kokonaislaskenta-ajat. DY-ratkaisijan kokonaislaskenta-aika 30 kierroksen jälkeen on 7,2 s, LU-hajotelman 2,9 s. LU-hajotelma on nyt tehokkaampi menetelmä. DY-ratkaisijalla kestää erityisen kauan saavuttaa ensimmäinen ratkaisu $\mathbf{x}_1(t_1)$.

Analysoidaan tarkemmin, mitä kuvassa 4.1 tapahtuu. Alussa (ajanhetkellä $t_0 = 0$) suhteellinen virhe on hieman yli 30. Ensimmäisen ratkaisun $\mathbf{x}_1(t_1)$ saavuttaminen on hyvin hidasta sekä systeemin ajan että laskenta-ajan kannalta. Tämä johtuu siitä, että alkutila $\mathbf{x}_{c,1}(t_0) = \text{randn}(n + p, 1)$ on huonosti valittu. Ensimmäinen ratkaisu on saavutettu ajanhetkellä $t_1 \approx 1400$, jolloin $\epsilon(t) < \epsilon_{\text{tol}}$. Ratkaisun suhteellinen

virhe on siis saatu pienemmäksi kuin ϵ_{tol}. Kun matriisia C häiritään, ratkaisun virhe nousee toleranssia ϵ_{tol} suuremmaksi mutta pysyy pienempänä kuin $2\epsilon_{\text{tol}}$, mikä on toivottua. Kun vanhan ratkaisun lopputila on asetettu häirityn systeemin alkutilaksi, uuden ratkaisun saavuttaminen on hyvin nopeaa sekä systeemin ajan että laskenta-ajan kannalta. Seuraava ratkaisu on saavutettu ajanhetkellä $t_2 \approx 1900$, jolloin virhe on jälleen pienempi kuin ϵ_{tol}. Samat havainnot ovat voimassa seuraavillakin kierroksilla. Systeemin häiritseminen kasvattaa virheen välille $(\epsilon_{\text{tol}}, 2\epsilon_{\text{tol}})$, ja asettamalla vanha lopputila uudeksi alkutilaksi uusi ratkaisu saavutetaan nopeasti. Neliömatriisin tapauksessa DY-ratkaisija on hitaampi kuin LU-hajotelma. Varsinkin ensimmäinen kierros oli hidas huonon alkutilavalinnan takia. LU-hajotelmalla laskenta-aika on joka kierroksella likimäärin vakio, mikä on järkevää, kun otetaan huomioon, että lineaarinen yhtälöryhmä ratkaistaan uudelleen joka kierroksella.

Lisäsimulaatioilla huomataan, että ensimmäinen kierros on heikon alkutilavalinnan takia aina DY-ratkaisijan hitain vaihe matriisin koosta riippumatta. Myös leveillä matriiseilla ensimmäinen kierros on hitain. Koska hyvää alkutilaa on käytännössä mahdoton valita satunnaisesti, haetaan DY-ratkaisijan ensimmäinen ratkaisu LU-hajotelman avulla. Tiedetään, että stabiilille systeemille $\mathbf{x}_c'(t) = A_c\mathbf{x}_c(t) + \mathbf{r}_c$ on voimassa, että $A_c\mathbf{x}_c(t)+\mathbf{r}_c \to \mathbf{0}$, kun $t \to \infty$. Kirjoitetaan tämä raja-arvo alkioittain:

$$\begin{bmatrix} -\alpha I_n & -k\alpha^2\widetilde{C}^* \\ C & O \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \xi \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ -\mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Saadaan yhtälöpari

$$\begin{cases} -\alpha\mathbf{x} - k\alpha^2\widetilde{C}^*\xi = \mathbf{0} \\ C\mathbf{x} - \mathbf{r} = \mathbf{0} \end{cases} \overset{\alpha>0}{\iff} \begin{cases} \mathbf{x} = -k\alpha\widetilde{C}^*\xi \\ C\mathbf{x} = \mathbf{r} \end{cases}. \tag{4.4}$$

Sijoittamalla $\mathbf{x} = -k\alpha\widetilde{C}^*\xi$ yhtälöön $C\mathbf{x} = \mathbf{r}$ saadaan

$$-k\alpha C\widetilde{C}^*\xi = \mathbf{r},$$

eli

$$C\widetilde{C}^*\xi = -\frac{1}{k\alpha}\mathbf{r}. \tag{4.5}$$

Yhtälön (4.5) ratkaisu voidaan laskea LU-hajotelman avulla. Kun ratkaisu ξ tiedetään, vektori $\mathbf{x}$ saadaan suoraan yhtälön (4.4) ensimmäisen rivin avulla: $\mathbf{x} = -k\alpha\widetilde{C}^*\xi$. Huomaa, että nyt LU-hajotelma tehdäänkin $p \times p$ -matriisille $C\widetilde{C}^*$ eikä $p \times n$ -matriisille C. Mikäli $p < n$, yhtälöryhmän (4.4) ratkaisu LU-hajotelman avulla voi olla jopa nopeampaa kuin yhtälön $C\mathbf{x} = \mathbf{r}$ ratkaisu LU-hajotelman avulla. Kun vektorit $\mathbf{x}$ ja ξ on saatu laskettua, asetetaan differentiaaliyhtälön ensimmäiseksi

ratkaisuksi

$$\mathbf{x}_{c,1}(t_1) = \begin{bmatrix} \mathbf{x} \\ \xi \end{bmatrix},$$

missä $t_1 = t_0 = 0$. Systeemin häiritsemisen jälkeen asetetaan ensimmäinen ratkaisu $\mathbf{x}_{c,1}(t_1)$ uudeksi alkutilaksi $\mathbf{x}_{c,2}(t_1)$. Häirityt systeemit ratkaistaan normaaliin tapaan DY-ratkaisijan avulla. Ainoastaan ensimmäisen kierroksen ratkaisumenetelmää muutettiin.

Suoritetaan vielä kerran DY-ratkaisijan ja LU-hajotelman laskenta-aikojen vertailu samalla matriisilla C ja samalla vektorilla $\mathbf{r}$. Ratkaistaan vektori $\mathbf{x}$ ja häiritään matriisia C 30 kierroksen ajan, mutta DY-ratkaisijalla käytetäänkin LU-hajotelmaa ensimmäisellä kierroksella, ja muilla kierroksilla käytetään DY-ratkaisijaa normaalisti. Ensimmäisellä kierroksella DY-ratkaisijan laskenta-aika saadaan mittaamalla se aika, joka kuluu yhtälön $C\widetilde{C}^*\xi = -\frac{1}{k\alpha}\mathbf{r}$ ratkaisuun sekä matriisi-vektoritulon $\mathbf{x} = -k\alpha\widetilde{C}^*\xi$ laskemiseen. Valitaan $\widetilde{C}^*$ jälleen C:n skaalausmatriisin avulla: $\widetilde{C}^* = C^{\sigma_1^2}$. Lineaarisen yhtälöryhmän ratkaisun suhteellisen virhe $\epsilon(t)$ ja menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.2.

Huomataan, että DY-ratkaisija on nyt melko tehokas ensimmäisellä kierroksella. Kuvan 4.2 ylemmästä kuvaajasta nähdään myös, että virhe on aluksi likimäärin 0, koska ensimmäinen ratkaisu haettiin LU-hajotelman avulla. Häiriön jälkeen virhe on kasvanut välille $(\epsilon_{\text{tol}}, 2\epsilon_{\text{tol}})$. Seuraavat kierrokset menevät täysin samalla periaatteella kuin kuvan 4.1 tapauksessa. Asettamalla vanha lopputila uudeksi alkutilaksi häirityn systeemin ratkaisu saavutetaan nopeasti. LU-hajotelma on kuitenkin edelleen nopeampi kuin DY-ratkaisija.

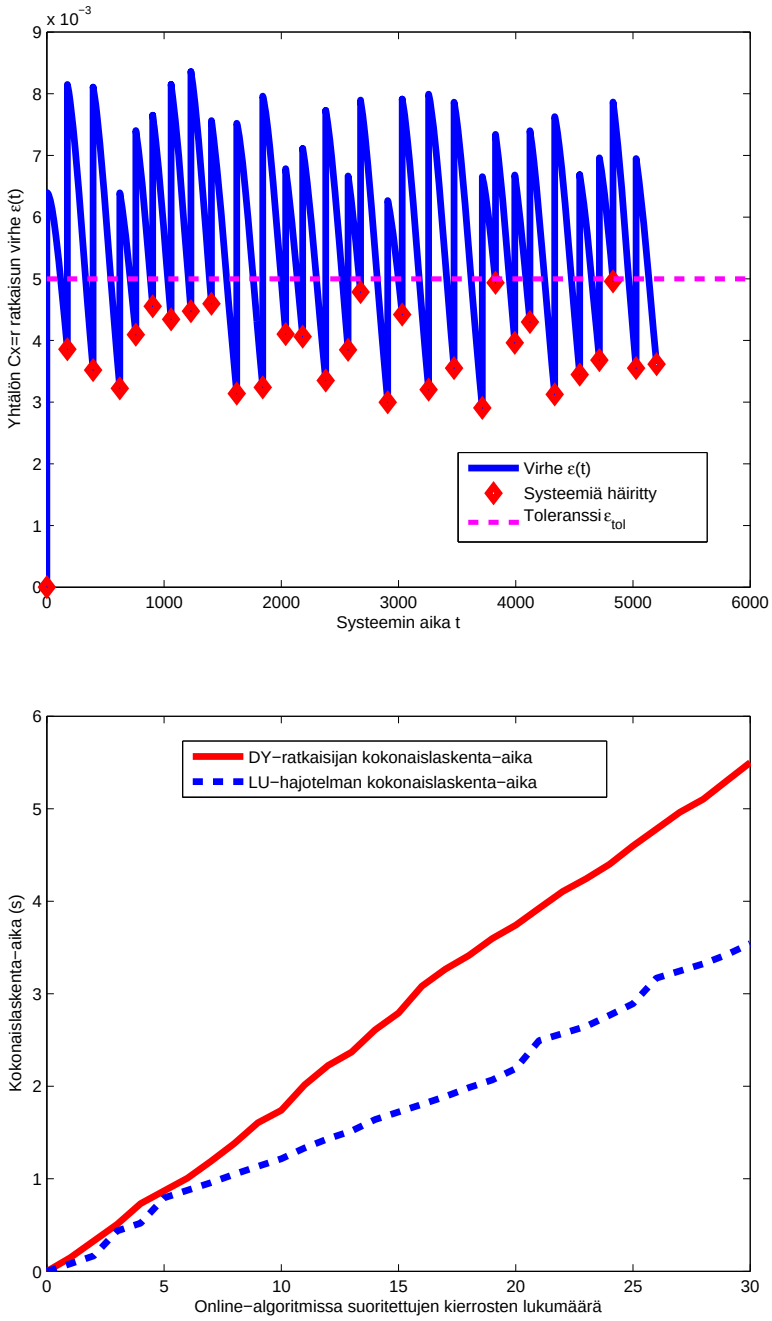
$\Diamond$

Neliömatriiseilla LU-hajotelma näyttäisi olevan tehokkaampi kuin DY-ratkaisija. Tärkeää oli kuitenkin saada DY-ratkaisijan hitain vaihe eli ensimmäinen kierros toimimaan laskennallisesti nopeasti. Haetaan jatkossakin DY-ratkaisijan ensimmäisen kierroksen ratkaisu LU-hajotelman avulla, ja muilla kierroksilla käytetään DY-ratkaisijaa normaaliin tapaan.

4.2.2 Leveät matriisit

Verrataan ratkaisijoita seuraavaksi tapauksissa, joissa C on leveä matriisi.

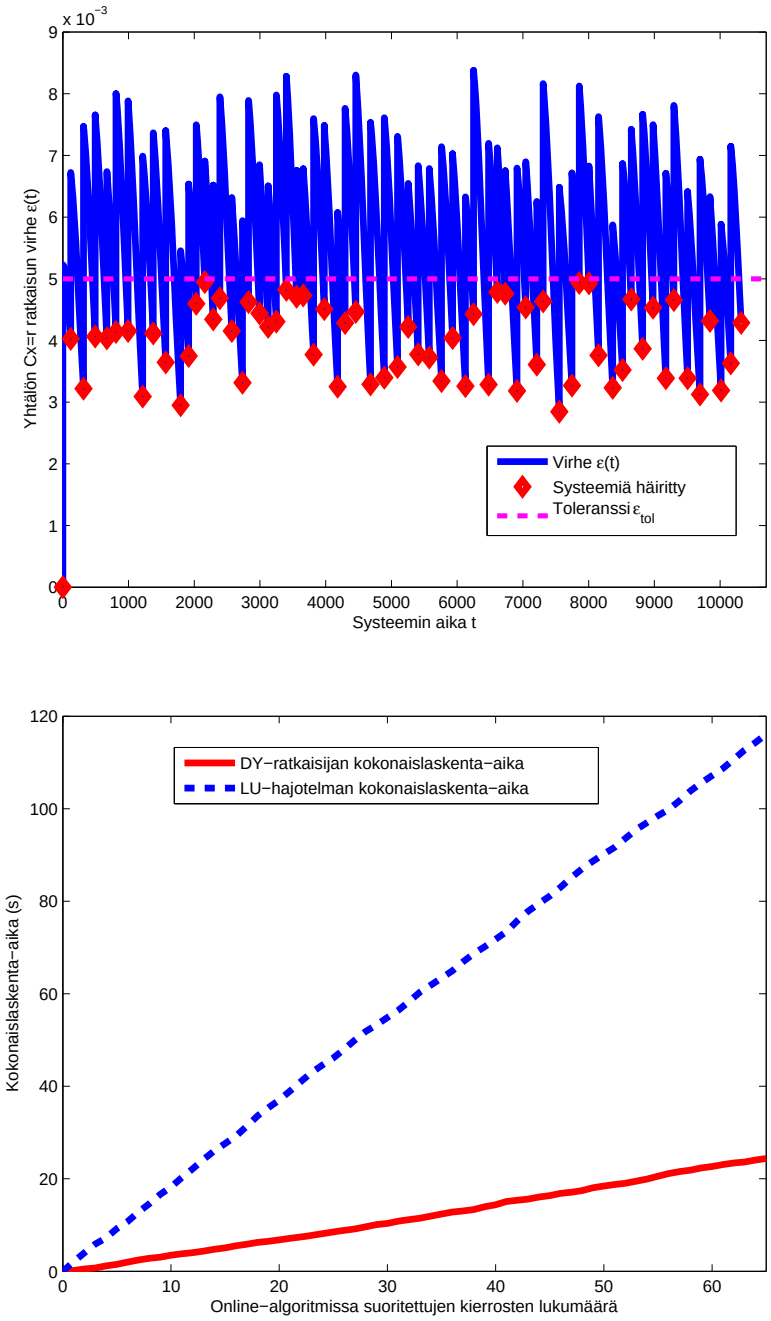
Esimerkki 4.2.2. Olkoon C 1063×1477 -matriisi ja $\mathbf{r}$ 1063×1 -vektori, ja oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Verrataan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja 65 kierroksen ajan, kun matriisia C häiritään. Differentiaaliyhtälön ratkaisemisessa valitaan $\widetilde{C}^*$ jälleen C:n skaalausmatriisiksi, eli asetetaan $\widetilde{C}^* = C^{\sigma_1^2}$. Hoidetaan DY-ratkaisijan ensimmäinen kierros



Kuva 4.2: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1000×1000 -matriisi ja ensimmäinen ratkaisu on kummassakin tapauksessa haettu LU-hajotelman avulla. Ylhäällä on suhteellisen virheen $\epsilon(t)$ kuvaaja, ja alhaalla on ratkaisijoiden kokonaislaskenta-ajat. DY-ratkaisijan kokonaislaskenta-aika 30 kierroksen jälkeen on 5,5 s, LU-hajotelman 3,5 s. LU-hajotelma on vieläkin tehokkaampi menetelmä, mutta ainakin DY-ratkaisija selviää jokaisesta kierroksesta melko tasaisesti.

LU-hajotelman avulla ja käytetään muilla kierroksilla DY-ratkaisijaa. Lineaarisen yhtälöryhmän ratkaisun suhteellinen virhe $\epsilon(t)$ sekä menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.3.

Ensimmäisellä kierroksella DY-ratkaisija on jopa nopeampi kuin LU-hajotelma,



Kuva 4.3: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1063×1477 -matriisi. Ylhäällä on suhteellisen virheen $\epsilon(t)$ kuvaaja, ja alhaalla on ratkaisijoiden kokonaislaskenta-ajat. DY-ratkaisijan kokonaislaskenta-aika 65 kierroksen jälkeen on 24,4 s, LU-hajotelman 116,1 s. DY-ratkaisija on nyt selvästi tehokkaampi kuin LU-hajotelma.

koska 1063×1063 -matriisin $CC^{\sigma_1^2}$ kääntäminen on huomattavasti nopeampaa kuin 1063×1477 -matriisin C kääntäminen. Muilla kierroksilla DY-ratkaisija päihittää LU-hajotelman siksi, että häirityn systeemin ratkaisu on lähellä uudeksi alkutilaksi valittua vanhan systeemin ratkaisua, kun häiriöt ovat riittävän pieniä. Kokonaislaskenta-aikojen kannalta DY-ratkaisija on jopa lähes 5 kertaa nopeampi

kuin LU-hajotelma. Jos mietitään esimerkin 4.2.1 kuvan 4.2 tilannetta, keskimääräiset laskenta-ajat yhtä kierrosta kohti olivat DY-ratkaisijalle 0,18 s ja LU-hajotelmalle 0,12 s. Tämän esimerkin tapauksessa keskimääräiset laskenta-ajat kierrosta kohti ovat puolestaan DY-ratkaisijalle 0,38 s ja LU-hajotelmalle 1,8 s. Kokeilemalla menetelmiä 1477×1477 -neliömatriisilla C saadaan keskimääräisiksi laskenta-ajoiksi kierrosta kohti 0,42 s DY-ratkaisijalle ja 0,31 s LU-hajotelmalle. Leveä matriisi hidastaa siis huomattavasti vain LU-hajotelmaa, mutta DY-ratkaisijan tehokkuus ei merkittävästi heikkene.

$\Diamond$

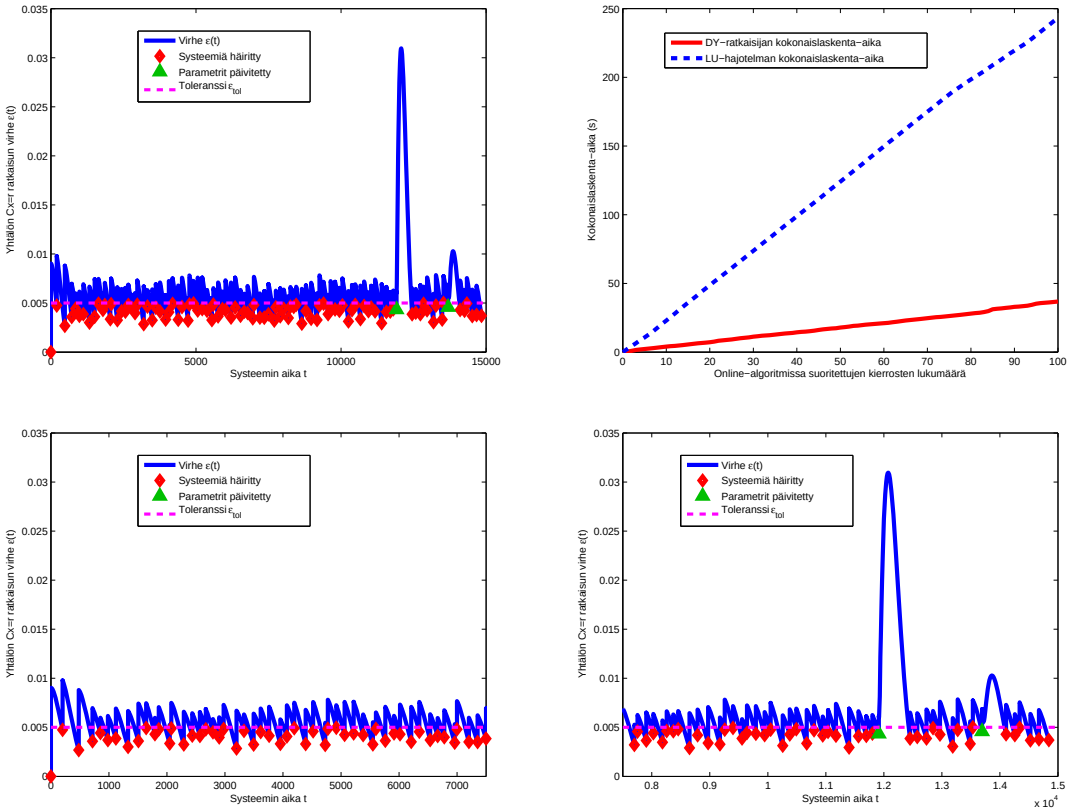
Levennetään matriisia C edelleen, jotta päästään kokeilemaan valinnan $\widetilde{C}^* = C^{\sigma_1^2}$ lisäksi myös vaihtoehtoa $\widetilde{C}^* = C^*$. Konjugaattitranspoosihan oli erittäin tehokas valinta $\widetilde{C}^*$:lle nimenomaan silloin, kun C on hyvin leveä matriisi. Kokeillaan myös sellaisia matriiseja, joiden alkiot eivät ole normaalijakautuneita. Vaihtelun vuoksi häiritään C:n lisäksi myös $\mathbf{r}$:ää.

Esimerkki 4.2.3. Olkoon C 1000×2000 -matriisi siten, että

$$C = \begin{bmatrix} \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix},$$

missä $\mathcal{C}_1$ ja $\mathcal{C}_2$ ovat hermiittisiä ($\mathcal{C}_i^* = \mathcal{C}_i$) 1000×1000 -matriiseja. Lisäksi olkoon $\mathbf{r}$ 1000×1 -vektori. Oletetaan, että $\mathcal{C}_1$:n alkiot noudattavat (jatkuvaa) tasajakaumaa $\mathrm{Tas}\left(-\frac{1}{10}, \frac{1}{10}\right)$, $\mathcal{C}_2$:n alkiot tasajakaumaa $\mathrm{Tas}(-1, 1)$ ja $\mathbf{r}$:n alkiot tasajakaumaa $\mathrm{Tas}(-10, 10)$. Verrataan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja 100 kierroksen ajan, ja häiritään sekä C:tä että $\mathbf{r}$:ää. Valitaan parametri $\widetilde{C}^*$ ensin C:n skaalausmatriisiksi: $\widetilde{C}^* = C^{\sigma_1^2}$. Haetaan differentiaaliyhtälön ensimmäinen ratkaisu jälleen LU-hajotelman avulla ja käytetään häirityillä systeemeillä DY-ratkaisijaa. Lineaarisen yhtölöryhmän ratkaisun suhteellinen virhe $\epsilon(t)$ ja menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.4.

DY-ratkaisija on jälleen huomattavasti nopeampi kuin LU-hajotelma. Myös ensimmäisellä kierroksella LU-hajotelma voitetaan. Häiriöt ovat sen verran pieniä, että ne eivät merkittävästi hidasta DY-ratkaisijaa, vaikka sekä C:tä että $\mathbf{r}$:ää häiritään. Ajanhetkillä $t \approx 11\,900$ ja $t \approx 13\,700$ systeemin parametrit α, k ja $\widetilde{C}^*$ on päivitetty. Parametrien päivittäminen muuttaa systeemiä ja siten differentiaaliyhtälön ratkaisua niin paljon, että lineaarisen yhtälöryhmän ratkaisun virhe kasvaa samalla huomattavasti. Laskenta-ajoissa tämä näkyy siten, että päivityksen jälkeinen ratkaisu on hieman hitaampi hakea kuin muut ratkaisut. Kuten kuvasta 4.4 kuitenkin nähdään, laskenta-aikojen hidastuminen on niin marginaalista, ettei päivityksen



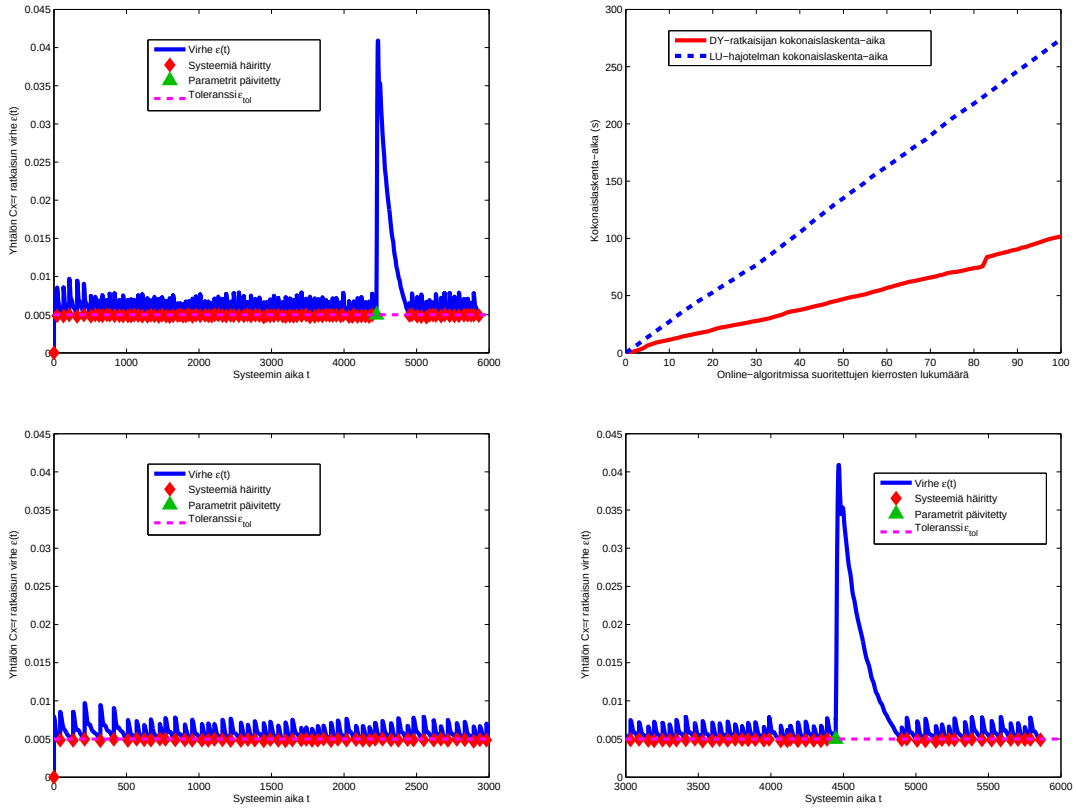
Kuva 4.4: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on kahdesta hermiittisestä lohkosta koostuva 1000×2000 -matriisi ja $\widetilde{C}^* = C^{\sigma_1^2}$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaaja suurennettuna ja jaettuna kahteen osaan. DY-ratkaisijan kokonaislaskenta-aika 100 kierroksen jälkeen on 36,8 s, LU-hajotelman 243,3 s. DY-ratkaisija on nyt selvästi tehokkaampi kuin LU-hajotelma.

jälkeisen ratkaisun hakemiseen tarvita lisätoimenpiteitä, kuten LU-hajotelmaa. Päivityskierroksella hävitty laskenta-aika saadaan muutenkin takaisin seuraavilla kierroksilla, sillä parametrien päivittäminen nopeuttaa DY-ratkaisijaa. Keskimääräiset laskenta-ajat yhtä kierrosta kohti ovat DY-ratkaisijalle 0,37 s ja LU-hajotelmalle 2,4 s. Kun verrataan edellisen esimerkin tapaukseen, jossa C oli 1063×1477 -matriisi, DY-ratkaisijan keskimääräinen laskenta-aika ei ole ollenkaan kasvanut, kun taas LU-hajotelmalla se on kasvanut yli puoli sekuntia.

Kuten kuvan 4.4 ylävasemmasta kuvaajasta nähdään, virheen $\epsilon(t)$ kuvaaja näyttää melko epäselvältä, jos koko 100 kierroksen simulaatio halutaan piirtää samaan kuvaajaan. Kuvan kahdessa alhaalla olevassa kuvaajassa virhe $\epsilon(t)$ on selkeyttämisen takia piirretty uudestaan alkuperäistä tarkemmalla skaalauksella. Alavasemmalla on $\epsilon(t)$ aikavälillä $0 \leq t < 7500$, ja alaoikealla $\epsilon(t)$ välillä $7500 \leq t < 15\,000$. Kun jatkossa häiriöiden määrää edelleen kasvatetaan, joudutaan virhe $\epsilon(t)$ esittämään yhä useammassa osassa, jotta kuvaaja olisi riittävän selkeä. Tällainen ei ole tarkoituksenmukaista. Jatkon kannalta toimitaan siten, että jos kuvaajaa ei pysty selkeästi esittämään korkeintaan kahdessa osassa, piirretään suurennos ainoastaan

ensimmäisestä 50 kierroksesta. Tällaisella kierrosmäärällä saa jo riittävän käsityksen funktion $\epsilon(t)$ käyttäytymisestä. Piirretään kuitenkin näkyviin myös virheen kuvaaja koko aikaskaalassa, koska heikosta selkeydestä huolimatta kuvaajasta nähdään ainakin $\epsilon(t)$:n muoto sekä päivitysten määrä ja tiheys.

Kokeillaan ratkaista differentiaaliyhtälö uudelleen käyttäen edelleen samaa C:tä ja $\mathbf{r}$:ää, mutta valitaan parametri $\widetilde{C}^*$ tällä kerralla C:n konjugaattitranspoosiksi: $\widetilde{C}^* = C^*$. Haetaan ensimmäinen ratkaisu LU-hajotelman avulla, ja muilla kierroksilla käytetään DY-ratkaisijaa. Lineaarisen yhtälöryhmän ratkaisun virhe $\epsilon(t)$ sekä menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.5.



Kuva 4.5: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on kahdesta hermiittisestä lohkosta koostuva 1000×2000 -matriisi ja $\widetilde{C}^* = C^*$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaaja suurennettuna ja jaettuna kahteen osaan. DY-ratkaisijan kokonaislaskenta-aika 100 kierroksen jälkeen on 101,8 s, LU-hajotelman 273,9 s. DY-ratkaisija on edelleen jokseenkin tehokkaampi kuin LU-hajotelma.

DY-ratkaisija päihittää vieläkin LU-hajotelman sekä kokonaislaskenta-ajassa että ensimmäisellä kierroksella. Parametrivalinnalla $\widetilde{C}^* = C^*$ DY-ratkaisija on kuitenkin huomattavasti hitaampi kuin asettamalla $\widetilde{C}^* = C\sigma_1^2$. Tämä johtuu luultavasti siitä, että matriisilla C on hyvin hankala rakenne: se koostuu kahdesta symmetrisestä lohkosta, joiden alkiot ovat lisäksi eri suuruusluokkaa. Matriisin C konjugaattitranspoosi on helppo muodostaa, mutta se ei yksinkertaista systeemiä kovin paljon. Tällöin hankalien matriisien tapauksessa laskenta pysyy työläänä ja laskenta-ajat

pitenevät. Sen sijaan skaalausmatriisi $C^{\sigma_1^2}$ muuttaa systeemin hyvin yksinkertaiseen muotoon, vaikka itse skaalausmatriisin laskeminen on hidasta singulaariarvohajotelman muodostamisen takia. Skaalausmatriisin hakemisen jälkeen laskenta onkin nopeaa. Jos kokeillaan verrata DY-ratkaisijaa ja LU-hajotelmaa 100 kierroksen ajan sellaisella matriisilla C, jonka alkiot ovat N(0, 1)-normaalijakautuneita, parametrivalinnalla $\widetilde{C}^* = C^*$ DY-ratkaisijan kokonaislaskenta-ajaksi saadaan noin 45 sekuntia — selvästi lyhyempi kuin tämän esimerkin matriisilla. LU-hajotelmalla laskenta-aika pysyy 225–250 sekunnissa, ja käyttämällä C:n skaalausmatriisia DY-ratkaisijan laskenta-aika on noin 35–40 sekuntia.

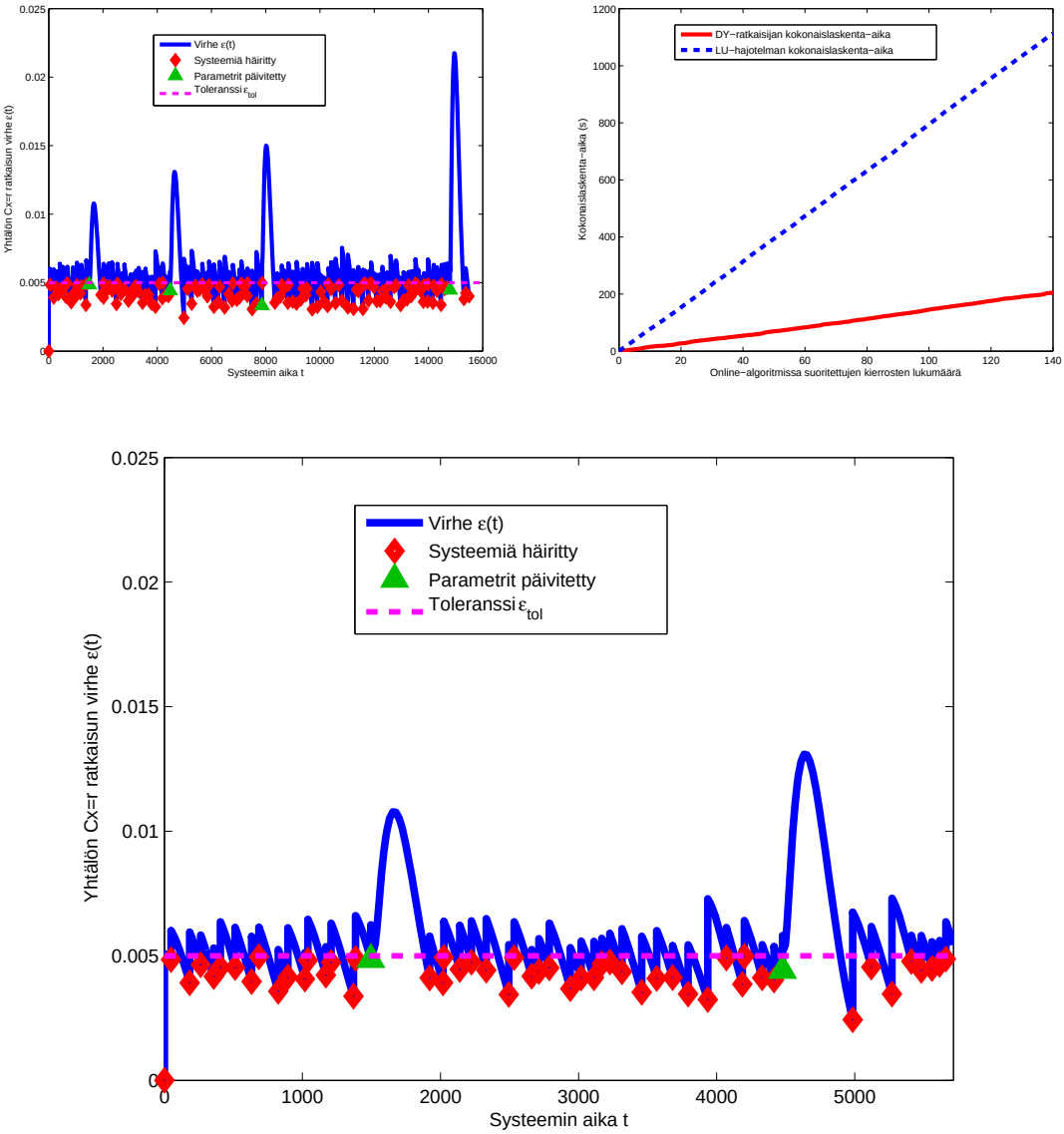
Kuvasta 4.5 huomataan, että systeemin parametrit α, k ja $\widetilde{C}^*$ päivitetään ajanhetkellä $t \approx 4400$. Päivitys kasvattaa ratkaisun virheen lähes 10-kertaiseksi sillä ajanhetkellä, ja seuraavalla kierroksella näkyy myös selvä nousu laskenta-aikojen kuvaajassa. Tämän jälkeen laskenta-ajat alkavat kuitenkin nopeutua. Keskimääräiset laskenta-ajat yhtä kierrosta kohti ovat DY-ratkaisijalle 1,0 s ja LU-hajotelmalle 2,7 s.

$\Diamond$

Matriisikoolla 1000×2000 parametrivalinta $\widetilde{C}^* = C^{\sigma_1^2}$ oli tehokkaampi kuin konjugaattitranspoosin käyttö valinnalla $\widetilde{C}^* = C^*$. Kokeillaan vieläkin leventää matriisia C, jolloin ainakin kuvan 3.16 perusteella konjugaattitranspoosin pitäisi toimia paremmin kuin skaalausmatriisin. Toisin kuin edellisessä esimerkissä, pidetään systeemi mahdollisimman yksinkertaisena.

Esimerkki 4.2.4. Olkoon C 1000×5000 -matriisi ja $\mathbf{r}$ 1000×1 -vektori, ja oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Verrataan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja 140 kierroksen ajan, kun matriisia C häiritään. Nyt siis vektori $\mathbf{r}$ pysyy samana koko ajan. Valitaan parametri $\widetilde{C}^*$ ensin C:n skaalausmatriisiksi: $\widetilde{C}^* = C^{\sigma_1^2}$. Differentiaaliyhtälön ratkaisussa hoidetaan ensimmäinen kierros LU-hajotelmalla ja muuten käytetään DY-ratkaisijaa. Lineaarisen yhtälöryhmän ratkaisun suhteellinen virhe $\epsilon(t)$ sekä menetelmien kokonaislaskenta-ajat on esitetty kuvassa 4.6.

DY-ratkaisija on jälleen yli 5 kertaa nopeampi kuin LU-hajotelma. Keskimääräiset laskenta-ajat kierrosta kohti ovat DY-ratkaisijalle 1,5 s ja LU-hajotelmalle 8,0 s. Leveällä matriisilla ratkaisu näyttäisi olevan vähemmän herkkä häiriöille kuin neliömatriisilla. Vaikka häiriöt onkin asetettu siten, että ratkaisun virhe on häiriön jälkeen välillä $(\epsilon_{\text{tol}}, 2\epsilon_{\text{tol}})$, kuvassa 4.6 häiriön jälkeinen virhe on keskimäärin selvästi lähempänä arvoa ϵ_{tol} kuin arvoa $2\epsilon_{\text{tol}}$. Kuten edellisessä esimerkissä, kuvan 4.6 ylävasemmassa kuvaajassa näkyy ratkaisun virheessä “piikki” jokaisen päivityksen kohdalla. Nämä piikit näyttävät voimistuvan jokaisen päivityksen kohdalla, eli

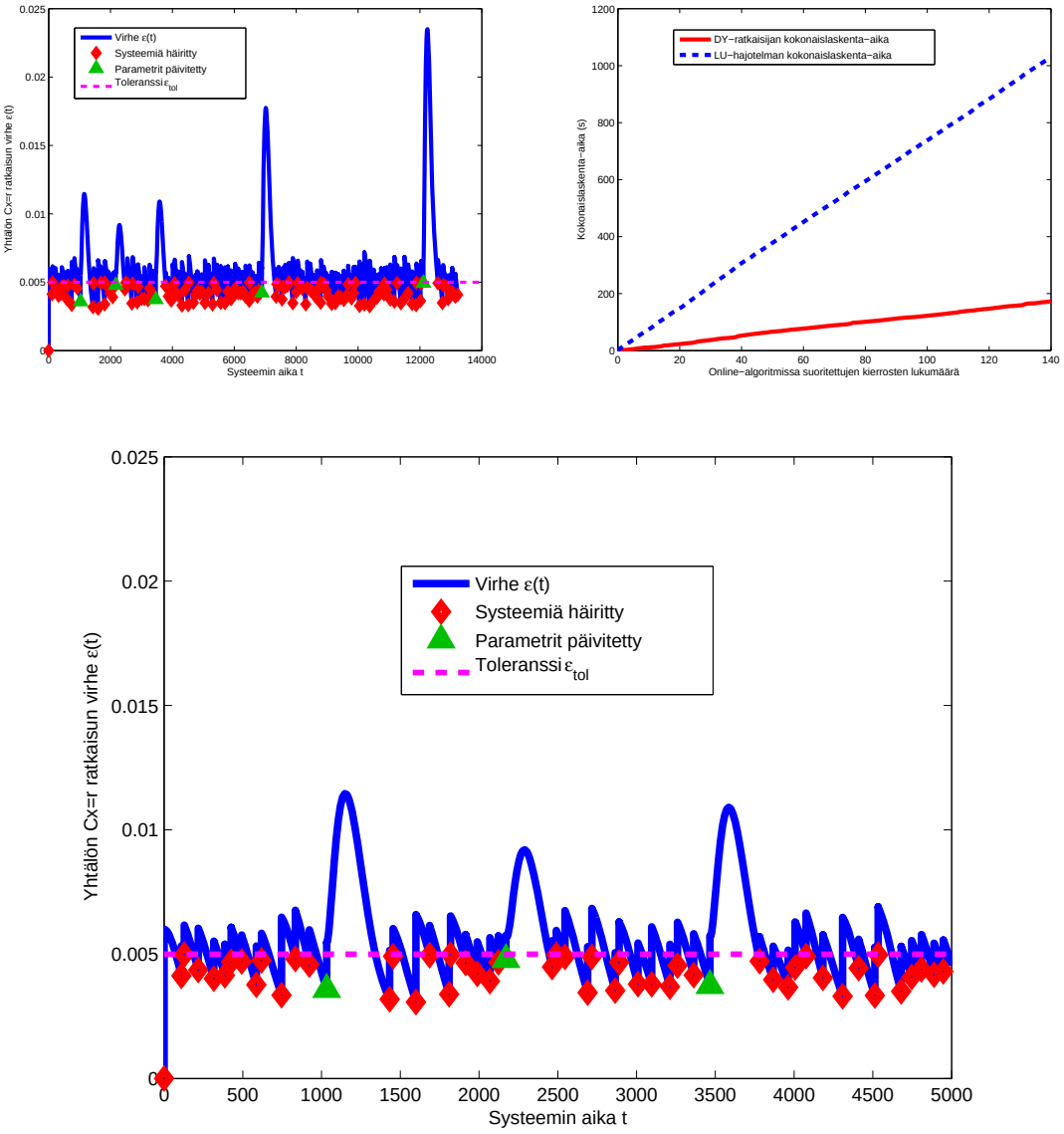


Kuva 4.6: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1000×5000 -matriisi ja $\widetilde{C}^* = C^{\sigma_1^2}$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaajan alkuosa suurennettuna. DY-ratkaisijan kokonaislaskenta-aika 140 kierroksen jälkeen on 205,5 s, LU-hajotelman 1113,1 s. DY-ratkaisija on selvästi tehokkaampi kuin LU-hajotelma.

kukin päivitys näyttäisi kasvattavan ratkaisun virhettä. Kyse on kuitenkin vain satunnaisesta ilmiöstä, joka ei ole yleisesti voimassa. Kasvattamalla vertailualgoritmin kierrosmäärää voidaan huomata, etteivät piikit aina suurene päivitysten myötä.

Palataan takaisin alkuperäiseen systeemiin ja verrataan laskenta-aikoja 140 kierroksen ajan, kun matriisia C häiritään. Muutetaan kuitenkin parametreja siten, että $\widetilde{C}^*$ valitaankin C:n konjugaattitranspoosiksi C^*. DY-ratkaisijan ja LU-hajotelman laskenta-aikoja tällä parametrivalinnalla on verrattu kuvassa 4.7. DY-ratkaisija on nytkin tehokkaampi kuin LU-hajotelma, ja edellinen on jopa hieman nopeampi kuin

kuvassa 4.6, kun käytetään parametrivalintaa $\widetilde{C}^* = C^*$. Keskimääräiset laskenta-ajat kierrosta kohti ovat DY-ratkaisijalle 1,2 s ja LU-hajotelmalle 7,4 s. Oleellista on kuitenkin huomata, että parametrivalinta $\widetilde{C}^* = C^*$ on käyttökelpoinen leveillä ja rakenteen kannalta mahdollisimman yksinkertaisilla matriiseilla.



Kuva 4.7: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1000×5000 -matriisi ja $\widetilde{C}^* = C^*$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaajan alkuosa suurennettuna. DY-ratkaisijan kokonaislaskenta-aika 140 kierroksen jälkeen on 172,8 s, LU-hajotelman 1030,5 s. DY-ratkaisija on selvästi nopeampi kuin LU-hajotelma, ja kuvaan 4.6 verrattuna konjugaattitranspoosi C^* on nyt tehokkaampi valinta parametrille $\widetilde{C}^*$ kuin skaalausmatriisi $C^{\sigma_1^2}$.

Kuvasta 4.7 voidaan tehdä samanlaisia havaintoja kuin kuvasta 4.6. Ratkaisun suhteellinen virhe häiriön jälkeen on edelleen melko pieni ja lähempänä arvoa ϵ_{tol} kuin arvoa $2\epsilon_{\text{tol}}$.

◊

4.2.3 Paluu neliömatriiseihin ja DY-ratkaisijan kehittäminen

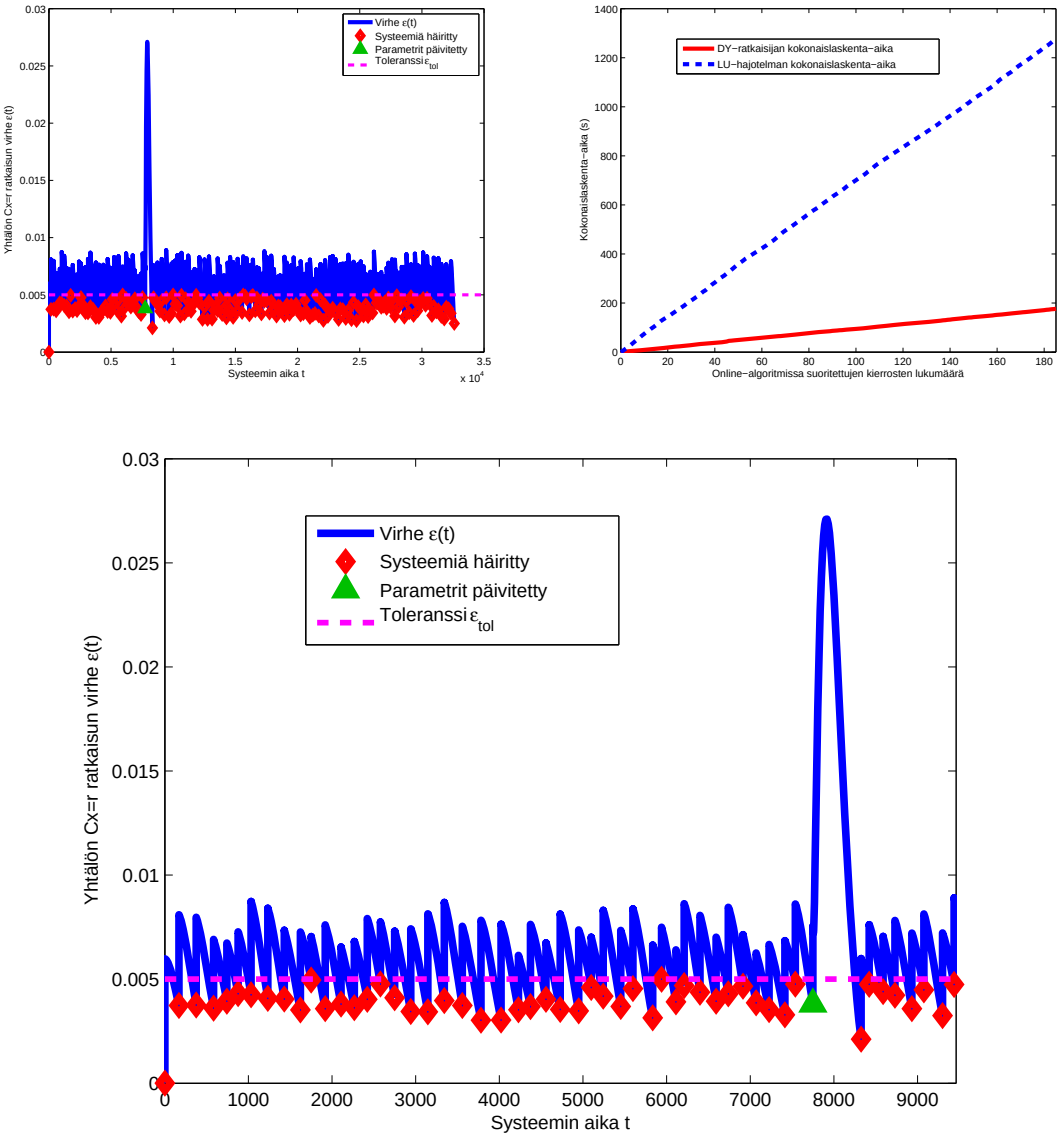
Tähän mennessä tarkastelluista esimerkeistä on havaittu, että DY-ratkaisija on silloin tehokkaampi kuin LU-hajotelma, kun C on leveä ja dimensioiltaan suuri matriisi. Tutkitaan seuraavaksi tapausta, jossa C on lähes neliömatriisi.

Esimerkki 4.2.5. Olkoon C 1999×2000 -matriisi ja $\mathbf{r}$ 1999×1 -vektori. Oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Verrataan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja 185 kierroksen ajan, kun C:tä häiritään ja $\mathbf{r}$ pysyy samana. Valitaan DY-ratkaisijan parametri $\widetilde{C}^*$ jälleen C:n skaalausmatriisiksi, eli $\widetilde{C}^* = C^{\sigma_1^2}$. Haetaan differentiaaliyhtälön ensimmäinen ratkaisu LU-hajotelman avulla ja muuten käytetään DY-ratkaisijaa. Lineaarisen yhtälöryhmän ratkaisun suhteellinen virhe $\epsilon(t)$ ja kokonaislaskenta-aikojen vertailu on esitetty kuvassa 4.8.

Huomataan, että C:n ollessa jopa lähes neliömatriisi DY-ratkaisija on selvästi tehokkaampi kuin LU-hajotelma. Laskenta-aikojen ero on merkittävän suuri: DY-ratkaisija on yli 7 kertaa nopeampi. Keskimääräiset laskenta-ajat yhtä kierrosta kohti ovat DY-ratkaisijalle 0,95 s ja LU-hajotelmalle 6,9 s. Suurilla $p \times n$ -matriiseilla ($n \geq p > 1000$) LU-hajotelma näyttäisi hidastuvan huomattavasti, jos $n > p$. Tämä johtuu todennäköisesti siitä, että leveän matriisin kääntäminen on laskennallisesti vaikeampaa kuin neliömatriisin, ja lisäksi matriisin suuri koko hidastaa kääntämistä. Tässä esimerkissähän $n = p + 1$, eli matriisin korkeus ja leveys poikkeavat toisistaan vain yhden dimension verran. Vaikutus laskenta-aikoihin on kuitenkin suuri. Kokeillaan vertailun vuoksi ratkaista lineaarinen yhtälöryhmä $C\mathbf{x} = \mathbf{r}$ LU-hajotelman avulla, kun C on neliömatriisi. Ajetaan simulaatio 10 000 erilaisella 2000×2000 -matrisiilla C ja 2000×1 -vektorilla $\mathbf{r}$ ja oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Yhtälöryhmän $C\mathbf{x} = \mathbf{r}$ ratkaisemiseen kuluva laskenta-aika on simulaation perusteella keskimäärin 0,67 s, mikä on selvästi nopeammin kuin 6,9 s. Itse asiassa LU-hajotelman 0,67 sekunnin keskimääräinen laskenta-aika 2000×2000 -matriisille on jopa nopeampi kuin DY-ratkaisijan 0,95 sekunnin keskimääräinen laskenta-aika 1999×2000 -matriisille.

◊

Tarkastellaan loppuhuipennuksena lineaarisen yhtälöryhmän ratkaisua hyvin suurella neliömatriisilla siten, että häiriöt ovat erittäin pieniä. Esimerkissä 4.2.1 LU-hajotelma oli 1000×1000 -neliömatriisin tapauksessa nopeampi kuin DY-ratkaisija.



Kuva 4.8: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 1999×2000 -matriisi. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaajan alkuosa suurennettuna. DY-ratkaisijan kokonaislaskenta-aika 185 kierroksen jälkeen on 176,6 s, LU-hajotelman 1277,8 s. Vaikka C onkin lähes neliömatriisi, DY-ratkaisija on silti huomattavasti tehokkaampi kuin LU-hajotelma.

Senkin jälkeen, kun differentiaaliyhtälön ensimmäinen ratkaisu haettiin LU-hajotelman avulla, kokonaislaskenta-ajat olivat 30 kierroksen jälkeen DY-ratkaisijalle 5,5 s ja LU-hajotelmalle 3,5 s. Jos C on neliömatriisi, onko olemassa sellaista tilannetta, jossa lineaarinen yhtälöryhmä $C\mathbf{x} = \mathbf{r}$ ratkeaa nopeammin DY-ratkaisijalla kuin LU-hajotelmalla?

Esimerkki 4.2.6. Olkoon C 4000×4000 -neliömatriisi ja $\mathbf{r}$ 4000×1 -vektori. Oletetaan, että C:n ja $\mathbf{r}$:n alkiot noudattavat N(0, 1)-normaalijakaumaa. Verrataan DY-

ratkaisijan ja LU-hajotelman laskenta-aikoja 1000 kierroksen ajan, kun sekä C:tä että $\mathbf{r}$:ää häiritään. Pienennetään vertailualgoritmissa häiriöitä kohdissa 5 ja 6 DY-ratkaisijan nopeuttamiseksi. Oletetaan, että ratkaisun virhe $\epsilon(t_i)$ on häiriön jälkeen välillä

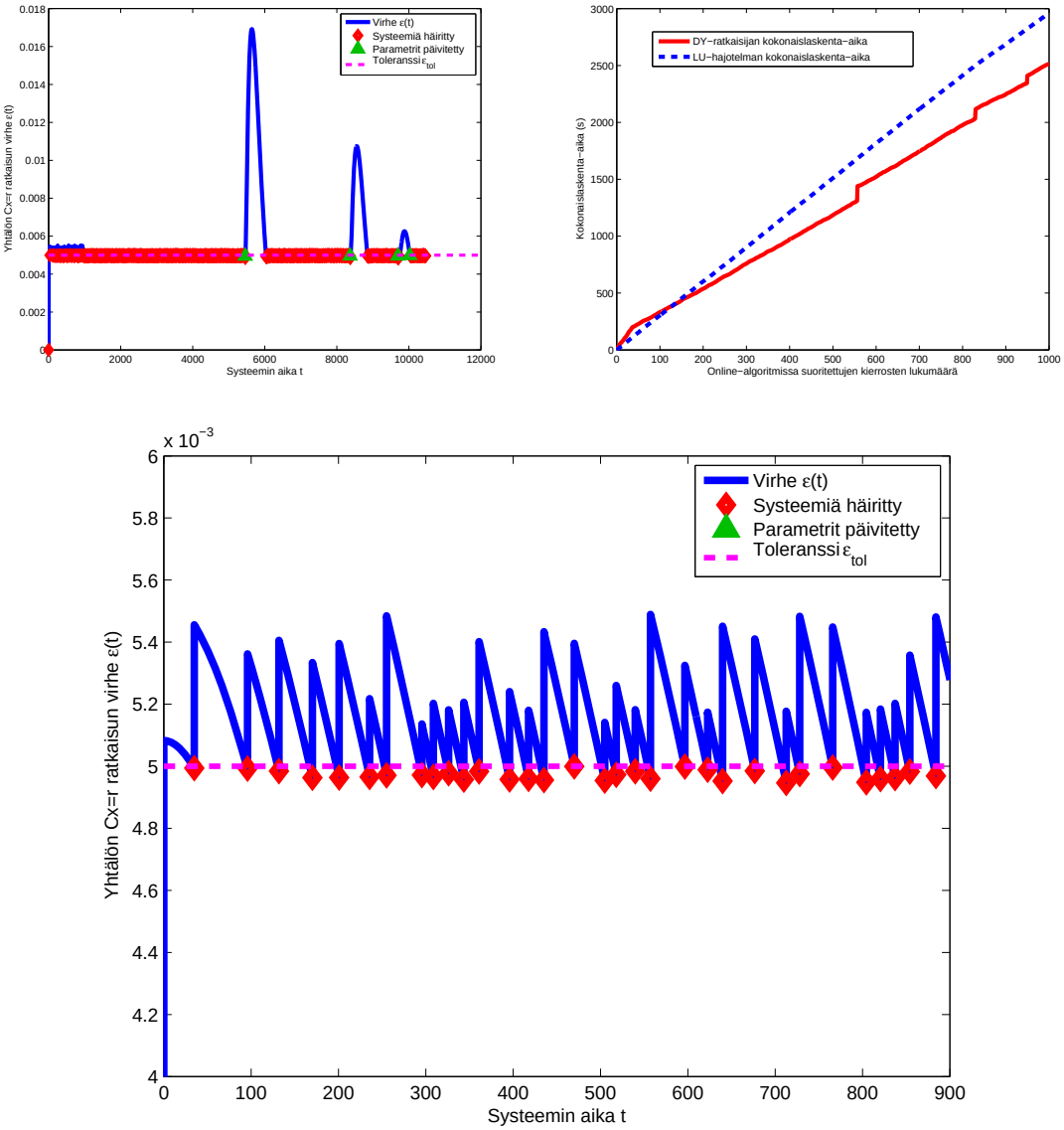
$$\epsilon_{\text{tol}} < \epsilon(t_i) < 1{,}1\epsilon_{\text{tol}},$$

missä $i = 1, 2, \ldots, 1000$ ja $\epsilon_{\text{tol}} = 5{,}0 \cdot 10^{-3}$. Häiriön jälkeinen virhe on siis välillä $5{,}0 \cdot 10^{-3} < \epsilon(t_i) < 5{,}5 \cdot 10^{-3}$.

Neliömatriiseilla C:n konjugaattitranspoosin käyttö ei ole tehokasta, joten hyödynnetään C:n singulaariarvohajotelmaa. Nyt matriisit ovat kuitenkin niin suuria, että singulaariarvohajotelman muodostaminen on hyvin hidasta. Pyritään siksi välttämään singulaariarvohajotelman muodostamista, ellei se ole laskenta-aikojen kannalta välttämätöntä. Singulaariarvohajotelma on laskettava ensimmäisellä kierroksella, kun parametri $\widetilde{C}^*$ alustetaan, ja se täytyy laskea uudelleen, kun parametreja päivitetään. Alustusta ei voi sivuuttaa, mutta päivitystiheyttä voidaan pienentää. Aikaisemmissa esimerkeissä parametrit päivitettiin, jos jonkin kierroksen laskenta-aika kasvoi yli 3,0-kertaiseksi edellisiin laskenta-aikoihin verrattuna. Nostetaan nyt rajaa siten, että parametrit päivitetään vain siinä tapauksessa, että laskenta-aika kasvaa yli 5,0-kertaiseksi edellisiin aikoihin verrattuna.

Valitaan DY-ratkaisijan parametri $\widetilde{C}^*$ vaihtelun vuoksi C:n pseudoinverssiksi, eli $\widetilde{C}^* = C^+$. Kuvasta 3.16 muistetaan, että pseudoinverssin käyttö on $p \times n$ -matriisin tapauksessa tehokasta silloin, kun $p \approx n$. Nyt on voimassa $p = n$, joten pseudoinverssin käyttö on perusteltavissa. Hoidetaan DY-ratkaisijan ensimmäinen kierros LU-hajotelman avulla. Lineaarisen yhtälöryhmän ratkaisun suhteellinen virhe $\epsilon(t)$ sekä kokonaislaskenta-aikojen vertailu on esitetty kuvassa 4.9.

Huomataan, että ensimmäistä kertaa neliömatriisien tapauksessa DY-ratkaisija on lievästi nopeampi kuin LU-hajotelma. Ensimmäisellä kierroksella LU-hajotelma on kuitenkin nopeampi, mikä johtuu siitä, että kummallakin menetelmällä ensimmäisen ratkaisun hakeminen vaatii 4000×4000 -matriisin kääntämistä, ja DY-ratkaisijalla on lisäksi laskettava matriisi-vektoritulo $\mathbf{x} = -k\alpha\widetilde{C}^*\xi$. Häirityillä systeemeillä DY-ratkaisija kuitenkin päihittää LU-hajotelman. Edellisen tehokkuus selittyy ennen kaikkea sillä, että C:n ja $\mathbf{r}$:n häiriöt ovat hyvin pieniä. Kuvan 4.9 alhaalla olevasta suurennetusta kuvaajasta huomataan, että virhe pysyy välillä $\epsilon_{\text{tol}} < \epsilon(t) < 1{,}1\epsilon_{\text{tol}}$. DY-ratkaisijan kokonaislaskenta-aika on 129. kierroksesta lähtien pienempi kuin LU-hajotelman. Tässä on yksi syy, miksi kierrosmäärä valittiin niinkin suureksi kuin 1000: laskenta-aikojen erot näkyvät vasta pitkällä aikavälillä monen häiriön jälkeen. DY-ratkaisijan parametrit on päivitetty yhteensä 4 kertaa. Kuvassa 4.9 päivitykset näkyvät “piikkeinä” virheen kuvaajassa. Toisin kuin kuvissa 4.6 ja 4.7, piikit eivät tässä esimerkissä kasvakaan päivitysten myötä. Laskenta-aikojen kuvaajassa on myös päivitysten kohdalla “hyppäys” DY-ratkaisijan tapauksessa. Keskimääräi-

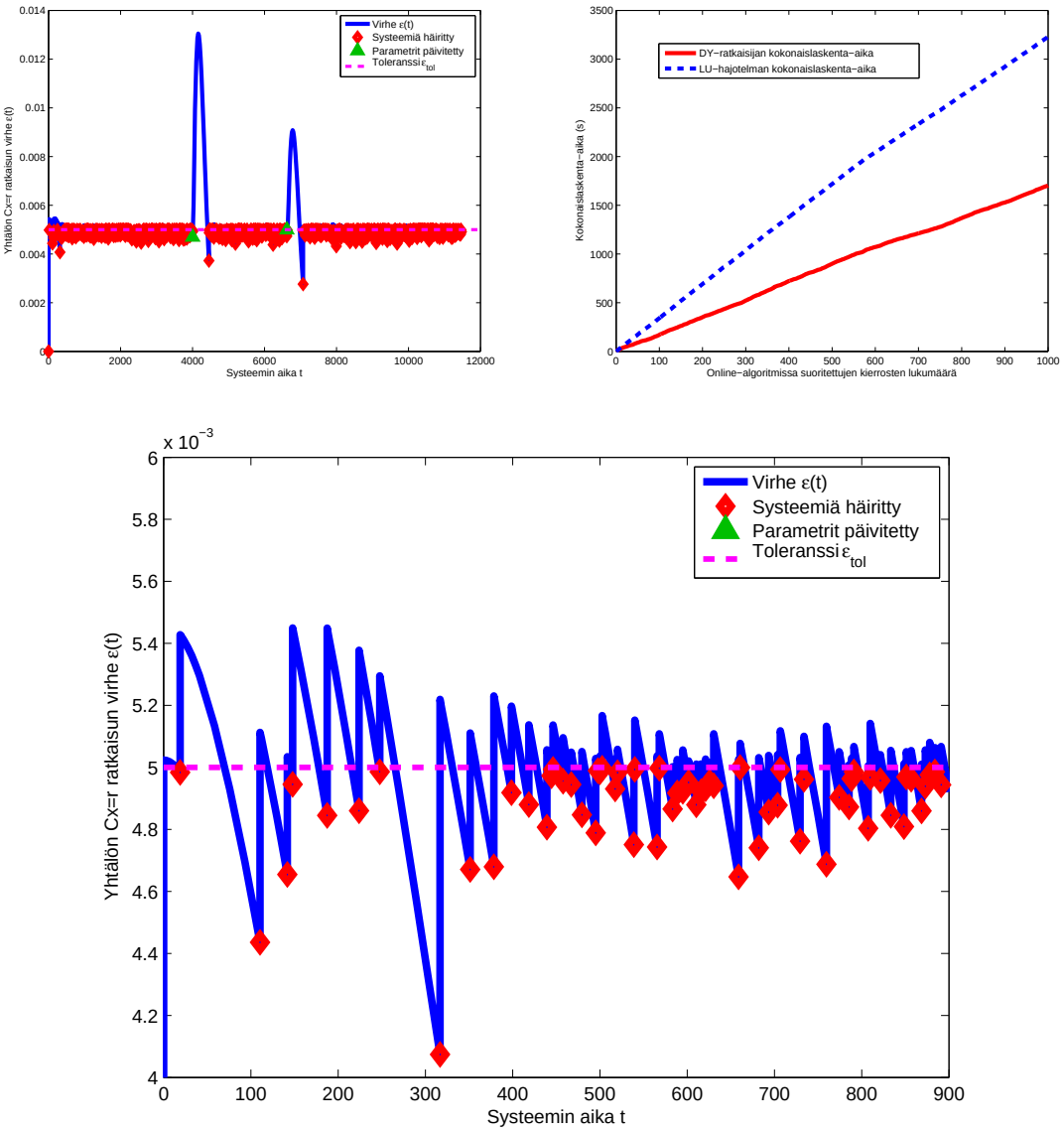


Kuva 4.9: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 4000×4000 -matriisi ja $\widetilde{C}^* = C^+$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaajan alkuosa suurennettuna. Huomaa, että suurennetussa kuvaajassa on eri skaalaus kuin ylävasemmassa kuvaajassa. DY-ratkaisijan kokonaislaskenta-aika 1000 kierroksen jälkeen on 2514,4 s, LU-hajotelman 2960,9 s. DY-ratkaisija on lievästi nopeampi kuin LU-hajotelma.

set laskenta-ajat kierrosta kohti ovat DY-ratkaisijalle 2,5 s ja LU-hajotelmalle 3,0 s. Edellisestä esimerkistä muistetaan, että LU-hajotelman keskimääräinen laskenta-aika 1999×2000 -matriisille oli 6,9 s. Tämäkin havainnollistaa sitä, miten LU-hajotelma hidastuu huomattavasti, kun C ei ole neliömatriisi.

Vaikka DY-ratkaisija onkin nyt nopeampi kuin LU-hajotelma, kokonaislaskenta-aikojen erot ovat hyvin pienet. Laskenta-ajoissa ei myöskään oteta huomioon sitä ylimääräistä työtä, joka singulaariarvohajotelman muodostamiseksi tehdään. Simulaa-

tion aikana singulaariarvohajotelma on muodostettu yhteensä 5 kertaa: ensimmäisellä kierroksella sekä 4 parametripäivityksen yhteydessä. Todellisuudessa singulaariarvohajotelman muodostamiseen vaadittava laskentatyö on erittäin suuri 4000×4000 -matriisilla. Olisi siis toivottavaa, että DY-ratkaisija päihittäisi LU-hajotelman suuremmalla laskenta-aikojen erolla kuin kuvassa 4.9. Palataan takaisin alkuperäiseen systeemiin ja toistetaan simulaatio, mutta valitaan parametri $\widetilde{C}^*$ tällä kerralla C:n skaalausmatriisiksi: $\widetilde{C}^* = C^{\sigma_1^2}$. DY-ratkaisijan ja LU-hajotelman laskenta-aikoja on verrattu kuvassa 4.10.



Kuva 4.10: DY-ratkaisijan ja LU-hajotelman vertailua, kun C on 4000×4000 -matriisi ja $\widetilde{C}^* = C^{\sigma_1^2}$. Ylävasemmalla on suhteellisen virheen $\epsilon(t)$ kuvaaja, yläoikealla ratkaisijoiden kokonaislaskenta-ajat. Alhaalla on virheen $\epsilon(t)$ kuvaajan alkuosa suurennettuna. Huomaa, että suurennetussa kuvaajassa on eri skaalaus kuin ylävasemmassa kuvaajassa. DY-ratkaisijan kokonaislaskenta-aika 1000 kierroksen jälkeen on 1703,5 s, LU-hajotelman 3228,8 s. DY-ratkaisija on nyt jonkin verran nopeampi kuin LU-hajotelma.

DY-ratkaisija on tehokkaampi kuin LU-hajotelma myös parametrivalinnalla $\widetilde{C}^* = C^{\sigma_1^2}$, ja tällä valinnalla laskenta-aikojen erot ovat paljon selvemmät kuin kuvassa 4.9. Ensimmäisellä kierroksella LU-hajotelma on nopeampi kuin DY-ratkaisija, mutta häirityillä systeemeillä DY-ratkaisija voittaa LU-hajotelman, kun C:n ja $\mathbf{r}$:n häiriöt ovat hyvin pieniä. Kuvan 4.10 alhaalla olevasta suurennetusta kuvaajasta huomataan, että virhe pysyy välillä $\epsilon_{\text{tol}} < \epsilon(t) < 1{,}1\epsilon_{\text{tol}}$. Häiriöiden suuruudessa näyttäisi olevan suurta vaihtelua. Tämä johtuu siitä, ensimmäisillä kierroksilla ratkaisun virhe pysyi käytössä olevilla häiriöillä halutulla välillä. Jossain vaiheessa häiriöt alkoivat kuitenkin olla liian suuria systeemille, joten häiriöitä pienennettiin. Neliömatriisit ovat sen verran herkkiä häiriöille, että sopivan suuruista häiriötä on vaikea muodostaa. Tyydytään tässä esimerkissä siihen, että virheet pysyvät häiriön jälkeen välillä $\epsilon_{\text{tol}} < \epsilon(t) < 1{,}1\epsilon_{\text{tol}}$, vaikka virhe ei olekaan tasaisesti jakautunut tälle välille. Häirityillä systeemeillä DY-ratkaisija on sen verran nopeampi kuin LU-hajotelma, että jo 5. kierroksesta lähtien edellisen kokonaislaskenta-aika on pienempi kuin jälkimmäisen. Parametrivalinnalla $\widetilde{C}^* = C^+$ LU-hajotelma ohitettiin vasta 129. kierroksella. DY-ratkaisijan parametrit on päivitetty yhteensä 2 kertaa. Kuvassa 4.10 päivitykset näkyvät jälleen "piikkeinä" virheen kuvaajassa, ajanhetkillä $t \approx 4200$ ja $t \approx 6800$. Keskimääräiset laskenta-ajat kierrosta kohti ovat DY-ratkaisijalle 1,7 s ja LU-hajotelmalle 3,2 s. DY-ratkaisija on siis tässä tapauksessa melkein 2 kertaa nopeampi kuin LU-hajotelma.

$\Diamond$

5. JOHTOPÄÄTÖKSET

Luvun 3 alussa on esitetty diplomityön tutkimuskysymys. Tavoitteena on ratkaista sarja lineaarisia yhtälöryhmiä $C_i\mathbf{x}_i = \mathbf{r}_i$, kun peräkkäisille yhtälöryhmille on voimassa $C_{i+1} \approx C_i$ ja $\mathbf{r}_{i+1} \approx \mathbf{r}_i$. Toisin sanottuna, matriisiin C_i ja vektoriin $\mathbf{r}_i$ kohdistetaan toistuvasti hyvin pieniä häiriöitä ja yhtälöryhmä ratkaistaan uudelleen jokaisen häiriön jälkeen. Tutkimuskysymyksen ratkaisemiseksi on esitetty DY-ratkaisija, jonka ideana on ratkaista yhtälöryhmä lineaarisen differentiaaliyhtälösysteemin avulla. Systeemin matriisi on muotoiltu siten, että differentiaaliyhtälö suppenee laskettavissa olevaa tasapainotilaa kohti, ja osa tilavektorin komponenteista suppenee lineaarisen yhtälöryhmän erästä ratkaisua kohti. Differentiaaliyhtälöä simuloidaan, kunnes yhtälöryhmän numeerinen ratkaisu saadaan laskettua.

DY-ratkaisija ei ole yksittäisten yhtälöryhmien ratkaisussa yhtä tehokas kuin LU-hajotelma, mutta menetelmää on pyritty kehittämään siten, että sillä voisi tehokkaasti ratkaista sarjan yhtälöryhmiä. Menetelmän vahvuutena on ratkaisijan robustisuus: yhtälöryhmän ratkaiseminen on mahdollista, vaikka kerroinmatriisia C_i ja oikean puolen vektoria $\mathbf{r}_i$ häirittäisiin. Kun tutkimuskysymys on ratkaistu teoreettisesti, voidaan alkaa kehittää DY-ratkaisijan parametreja. Kappaleissa 3.4 ja 3.5 on simulaatioiden avulla pyritty optimoimaan käytössä olevia parametreja. Lopuksi kappaleessa 3.6 on esitetty niin sanottu online-algoritmi, jossa menetelmällä ratkaistaan sarja lineaarisia yhtälöryhmiä.

Luvussa 4 on ensin hahmoteltu, miten DY-ratkaisijaa ja LU-hajotelmaa voisi verrata keskenään. Matriisin C_i ja vektorin $\mathbf{r}_i$ häiriöt on valittu tutkimuskysymyksen oletusten mukaisesti pieniksi. Kappaleessa 4.1.3 on tarkasteltu, millä tavalla ja missä tilanteissa DY-ratkaisijan parametreja päivitetään tarvittaessa. Seuraavaksi on koottu algoritmi, jossa sekä DY-ratkaisijalla että LU-hajotelmalla ratkaistaan N kappaletta lineaarisia yhtälöryhmiä, ja kummankin menetelmän laskenta-aikoja verrataan samalla.

Kappaleen 4.2 esimerkeissä on verrattu DY-ratkaisijan ja LU-hajotelman laskenta-aikoja suurilla matriiseilla. Osa matriiseista on ollut leveitä, ja osa neliömatriiseja. Matriisit ovat olleet suuria laskenta-aikojen kasvattamiseksi, koska liian pienillä laskenta-ajoilla menetelmien vertailu olisi ollut vaikeaa. Häiriöt on pidetty tutkimuskysymyksen mukaisesti pieninä, ja tällöin DY-ratkaisija saavuttaa nopeasti seuraavan ratkaisun hyödyntämällä edellistä ratkaisua alkuehtona. Mikäli häiriöt ovat liian

suuria, DY-ratkaisija hidastuu huomattavasti. Jos kokoa $p \times n$ oleva kerroinmatriisi on jotain muuta kuin neliömatriisi, eli $p < n$, DY-ratkaisija on tavallisesti nopeampi kuin LU-hajotelma. Mitä leveämpi C on, sitä suuremmalla laskenta-aikojen erolla LU-hajotelma häviää DY-ratkaisijalle. Tavallisesti tehokkain valinta parametrille $\widetilde{C}^*$ on C:n skaalausmatriisi $C^{\sigma_1^2}$. Sellaisilla leveillä matriiseilla, jotka ovat mahdollisimman yksinkertaisia, myös konjugaattitranspoosi C^* on järkevä valinta parametrille $\widetilde{C}^*$. Neliömatriiseilla voidaan vaihtoehtoisesti kokeilla parametrivalintaa $\widetilde{C}^* = C^+$, missä C^+ on C:n pseudoinverssi.

DY-ratkaisija voi toisinaan olla tehokkaampi kuin LU-hajotelma myös neliömatriisien tapauksessa, mutta tämä on mahdollista vain silloin, kun häiriöt ovat hyvin pieniä. Esimerkissä 4.2.6 DY-ratkaisija oli tehokkaampi menetelmä 4000×4000 -neliömatriisin tapauksessa. Jos häiriöt ovat riittävän pieniä, on mahdollista, että DY-ratkaisija päihittää LU-hajotelman myös kokoa 4000×4000 selvästi pienemmillä neliömatriiseilla. Esimerkissä 4.2.1 DY-ratkaisijan olisi ehkä voinut saada tehokkaammaksi kuin LU-hajotelma käyttämällä hyvin pieniä häiriöitä. Kuvan 4.2 tuloksista kuitenkin huomataan, että LU-hajotelman keskimääräinen laskenta-aika kierrosta kohti oli 0,12 s. Tällainen aika on jo niin pieni, ettei menetelmiä välttämättä pysty kovin tarkasti vertailemaan, sillä laskentakoneen satunnaiset kuormitukset vaikuttavat näissä tapauksissa huomattavasti aikoihin. Olennaista on, että DY-ratkaisija on leveillä matriiseilla tyypillisesti tehokkaampi kuin LU-hajotelma ja että joskus myös neliömatriiseilla LU-hajotelma voitetaan.

5.1 Jatkokehitysideoita

DY-ratkaisijan parametrien α, k ja $\widetilde{C}^*$ hienosäätöä voidaan edelleen jatkaa, jotta differentiaaliyhtälösysteemien ratkaisua saataisiin vielä nopeutettua. Toleranssin ϵ_{tol} vaikutusta DY-ratkaisijan laskenta-aikoihin voisi tutkia. Häiriöiden suuruutta ja jakaumaa voisi tutkia reaalimaailman sovelluksissa, jotta tiedettäisiin, millaisia häiriöt todellisissa tilanteissa ovat. Lisätutkimusta tarvittaisiin myös sille, millä ehdoilla parametreja kannattaa päivittää häiriöiden jälkeen. Tähän mennessä ainoana ehtona on ollut laskenta-aikojen hidastuminen. Myös parametrien päivittäminen vaatii ylimääräistä laskentaa, varsinkin jos parametrin $\widetilde{C}^*$ valinnassa käytetään C:n singulaariarvohajotelmaa. Olisi tärkeää löytää sopiva tasapaino, jotta parametreja ei päivitettäisi liian usein ja jotta häiriöt eivät toisaalta pääsisi hidastamaan laskenta-aikoja liian paljon.

Neliömatriisit ovat edelleen DY-ratkaisijan kannalta vaikeimmat tapaukset, joskin esimerkissä 4.2.6 DY-ratkaisija olikin tehokkaampi kuin LU-hajotelma. Tehokkainta tapaa ratkaista toistuvasti lineaarinen yhtälöryhmä $C_i\mathbf{x}_i = \mathbf{r}_i$, missä C_i on neliömatriisi ja $C_{i+1} \approx C_i$, voitaisiin siis edelleen tutkia. DY-ratkaisija on myös hidas silloin, kun edellisiä ratkaisuja ei tiedetä, koska tällöin on käytännössä mah-

dotonta valita alkutila järkevästi. Sopivan alkutilan hakeminen on myös yksi jatkotutkimusaihe. Tässä opinnäytetyössä ensimmäisen ratkaisun alkutila saatiin ratkaisemalla LU-hajotelmalla systeemin tasapainopisteet. Voisi myös pohtia, kannattaisiko parametrien päivityksen jälkeinen ratkaisu hakea esimerkiksi LU-hajotelman avulla, koska systeemin päivitys muuttaa differentiaaliyhtälön seuraavaa ratkaisua huomattavasti. Tällöin kyseisen ratkaisun laskeminen on hidasta. Tähän mennessä DY-ratkaisijaa on verrattu ainoastaan LU-hajotelman kanssa. Vertailu muiden menetelmien — kuten numeeristen ratkaisumenetelmien — kanssa olisi myös mahdollinen aihe jatkotutkimukselle. Lisäksi DY-ratkaisijaa voisi yrittää laajentaa siten, että algoritmi soveltuisi ∞-ulotteisten lineaaristen yhtälöiden ratkaisuun.

LÄHTEET

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney ja D. Sorensen. *LAPACK User's Guide*. 3. painos, SIAM, Philadelphia, 1999.
http://www.netlib.org/lapack/lug/lapack_lug.html
(viitattu 10.8.2011)

[2] R. E. Bank, W. C. Coughran Jr., W. Fichtner, E. Grosse, D. Rose ja R.Smith. *Transient Simulation of Silicon Devices and Circuits*. IEEE Trans. CAD, vol. 4, s. 436–451, 1985.

[3] P. Bogacki ja L. F. Shampine. *A 3(2) pair of Runge-Kutta formulas*. Appl. Math. Letters, vol. 2, s. 1–9, 1989.

[4] T. A. Davis. *UMFPACK Version 4.6 User Guide*. Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, 2002.
http://www.cise.ufl.edu/research/sparse/umfpack
(viitattu 10.8.2011)

[5] E. J. Davison. *The robust control of a servomechanism problem for linear time-invariant multivariable systems*. IEEE Trans. Autom. Contr., vol. AC-21, no. 1, s. 25–34, 1976.

[6] J. R. Dormand ja P. J. Prince. *A family of embedded Runge-Kutta formulae*. J. Comp. Appl. Math., vol. 6, s. 19–26, 1980.

[7] G. H. Golub ja C. F. Van Loan. *Matrix Computations*. 3. painos, Baltimore: Johns Hopkins Studies in Mathematical Sciences, ISBN 978-0-8018-5414-9, 1996.

[8] D. Hilbert. *Ein Beitrag zur Theorie des Legendre'schen Polynoms*. Acta Mathematica, vol. 18, s. 155–159, 1894.

[9] M. W. Hirsch, S. Smale ja R. L. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. 2. painos, Elsevier Academic Press, 2004.

[10] Imperial College London. *Matrix Manual: Matrix Property Proofs*. Department of Electrical and Electronic Engineering.
http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/proof003.html#DetBlock_p
(viitattu 25.5.2011)

[11] T. Kailath. *Linear Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[12] R. E. Kalman. *Contributions to the theory of optimal control*. Boletin de la Sociedad Matematica Mexicana, vol. 5, s. 102–119, 1960.

[13] J. Kauhanen. *MAT-10422 Insinöörimatematiikka B 2u, Periodi 2/2010–2011*. Opintomoniste, TTY, Matematiikan laitos, 2010.

[14] J. Kautsky, N. K. Nichols ja P. Van Dooren. *Robust Pole Assignment in Linear State Feedback*. Int. J. Control, vol. 41, no. 5, s. 1129–1155, 1985.

[15] H. W. Knobloch ja H. Kwakernaak. *Lineare Kontrolltheorie*. Akademie Verlag, Berliini, 1986.

[16] C. L. Lawson ja R. J. Hanson. *Solving Least Squares Problems*. Englewood Cliffs, NJ: Prentice-Hall, ISBN 0138225850, 1974.

[17] *MATLAB*. Versio 7.9.0.529 (R2009b). The MathWorks Inc., 2009.

[18] C. B. Moler ja C. F. Van Loan. *Nineteen Dubious Ways to Compute the Exponential of a Matrix*. SIAM Review, vol. 20, no. 4, s. 801–836, 1978.

[19] A. Perttula, K. Vattulainen ja T. Suurhasko. *Todennäköisyyslaskenta*. Opintomoniste kurssille MAT-20501 Todennäköisyyslaskenta, TTY, Matematiikan laitos, versio 9, 2010.

[20] PlanetMath. *PlanetMath: proof of block determinants*. http://planetmath.org/encyclopedia/ProofOfBlockDeterminants.html (viitattu 25.5.2011)

[21] S. A. Pohjolainen. *MAT-31090 Matriisilaskenta 1*. Opintomoniste, TTY, Matematiikan laitos, 2005.

[22] S. A. Pohjolainen. *MAT-31090 Matriisilaskenta 1*. Opintomoniste, TTY, Matematiikan laitos, 2008.

[23] S. A. Pohjolainen. *Robust multivariable PI-controller for infinite dimensional systems*. IEEE Trans. Autom. Contr., vol. AC-27, no. 1, s. 17–30, 1982.

[24] L. F. Shampine ja M. K. Gordon. *Computer Solution of Ordinary Differential Equations: the Initial Value Problem*. W. H. Freeman, San Francisco, 1975.

[25] L. F. Shampine ja M. E. Hosea. *Analysis and Implementation of TR-BDF2*. Applied Numerical Mathematics 20, 1996.

[26] L. F. Shampine ja M. W. Reichelt. *The MATLAB ODE Suite*. SIAM Journal on Scientific Computing, vol. 18, s. 1–22, 1997.

[27] L. F.Shampine, M. W. Reichelt ja J. A. Kierzenka. *Solving Index-1 DAEs in MATLAB and Simulink*. SIAM Review, vol. 41, s. 538–552, 1999.

[28] R. Silvennoinen. *Laaja matematiikka*. Opintomoniste, TTY, Matematiikan laitos, 2005.

[29] D. Simon. *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[30] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. 2. painos, Texts in Applied Mathematics, vol. 6, Springer, New York, 1998.

[31] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1965.

A. LIITTEITÄ

A.1 MATLAB-koodeja ratkaisijoiden laskenta-aikojen vertailuun

A.1.1 Systeemi.m

```
% Muodostaa systeemin x_c'(t) = A_c x_c(t) + r_c ode-ratkaisijaa varten

function dxc = Systeemi(t, xc, AC, rc, C)

dxc = AC*xc+rc;
```

A.1.2 lopehto.m

```
% Lopetusehtofunktio: DY-ratkaisijan toiminta lopetetaan,
% kun ratkaisun virhe on tiettyä toleranssia pienempi
% (tällöin muuttuja status saa arvon true)

function status = lopehto(t,xc,~,AC,rc,C)

epsilon = 0.5E-2;
n=size(C,2);
status = false;
if ( ~isempty(xc) )
    x = xc(1:n,end);
    r = -1*rc(n+1:end);
    erotus = norm(r - C*x) / norm(r);
    if ( erotus < epsilon )
        status = true;
    end
else
    status = true;
end
```

A.1.3 HaeErotus.m

```
% Lasketaan suhteellinen virhe erotukselle C*x(t) - r

function erotus = HaeErotus(t,xc,C,rc)

n=size(C,2);
```

```
erotus=zeros(size(t));
for i = 1:length(t)
    x = xc(i, 1:n)';
    r = -1*rc(n+1:end);
    erotus(i) = norm(r - C*x) / norm(r);
end
```

A.1.4 Vertailu.m

```
% Vertaillaan DY-ratkaisijan ja LU-hajotelman laskenta-aikoja,
% kun halutaan toistuvasti ratkaista yhtälö Cx = r.

max_kierroksia = 50;
aika_dy = 0;            % DY-ratkaisijan kokonaislaskenta-aika
aika_lu = 0;            % LU-hajotelman kokonaislaskenta-aika
T = 1E9;

C = randn(700,1000);
[p,n] = size(C);
v = svd(C);
r = randn(p,1);

% DY-ratkaisijan parametrien muotoilu
a = 0.01;
A = -a*eye(n);
Ctilde = v(1)^2*pinv(C);
k = 0.475/v(1)^2;
B = - k*a^2*Ctilde;
D = zeros(p);

% Kirjattavat asiat
aikaskaala = [];
virhe = [];
hairiokohta = [];
t1 = 0;
aikakirjaus = zeros(1,3);

% Yhtälöryhmä ratkaistaan toistuvasti max_kierroksia kertaa
for kierros = 1:max_kierroksia

    % Differentiaaliyhtälösysteemi
    % ---------------------------

    AC = [A B; C D];
    rc = [ zeros(n,1); -r ];

    % Ensimmäisellä kierroksella haetaan ratkaisu LU:lla
```

```
if ( kierros == 1 )
    tic
    xi_odealku = -(C*Ctilde)\r/(a*k);
    x_odealku = -a*k*Ctilde*xi_odealku;
    x1 = [x_odealku; xi_odealku]';
    aika=toc;
    t=0;
    erotusvek = HaeErotus(t,x1,C,rc);
    clear x_odealku xi_odealku
else
    options = odeset('RelTol', 1E-3, 'AbsTol', 1E-6, ...
        'Jacobian', AC, 'OutputFcn', @lopehto);
    tic
    [t,M]=ode45(@Systeemi, [t1, t1+T], x1, options, AC, rc, C);
    aika=toc;
    erotusvek = HaeErotus(t,M,C,rc);

    % Seuraavaa kierrosta varten loppuaika alkuajaksi ja
    % lopputila alkutilaksi
    t1 = t(end);
    x1 = M(end,:);
end

aika_dy = aika_dy + aika;


% LU-hajotelma
% ------------

tic
x_lu = C\r;
aika2 = toc;
aika_lu = aika_lu + aika2;


% Laskenta-aikojen kirjaus ja tulostus
aikaskaala = [aikaskaala; t];
virhe = [virhe; erotusvek];
hairiokohta = [hairiokohta; aikaskaala(end), virhe(end)];

if ( kierros == 1 )
    disp('[ Kierros | DY:n laskenta-aika | LU:n laskenta-aika ]')
end
disp([num2str(kierros), '   ', num2str(aika), ...,
    '    ', num2str(aika2)]);

% Kokonaislaskenta-aika tähän kierrokseen asti
```

```
    aikakirjaus = [aikakirjaus; kierros, ...
        aikakirjaus(end,2)+aika, aikakirjaus(end,3)+aika2];

    % Häiritään matriisia C
    C = C + 0.4E-2*randn(size(C));

end

disp(['Kokonaislaskenta-ajat: DY - ', num2str(aika_dy), ...
    ', LU - ', num2str(aika_lu)]);

% Kuvaaja DY-ratkaisijan virheestä
figure
plot(aikaskaala, virhe, 'LineWidth', 4)
hold on
plot(hairiokohta(:,1), hairiokohta(:,2), 'rd', 'LineWidth', 4)
x_akseli = get(gca, 'XLim');
epsilon = 0.5E-2;
plot( linspace(x_akseli(1), x_akseli(2)) , epsilon*linspace(1,1), ...
    'm--', 'LineWidth', 3)
xlabel('Systeemin aika t')
ylabel('Yhtälön Cx=r ratkaisun virhe \epsilon(t)')
legend('Virhe \epsilon(t)', 'Systeemiä häiritty', ...
    'Toleranssi \epsilon_{tol}')
hold off

% Kuvaaja laskenta-ajoista
figure
plot(aikakirjaus(:,1), aikakirjaus(:,2), 'r-', 'LineWidth', 4)
hold on
plot(aikakirjaus(:,1), aikakirjaus(:,3), 'b--', 'LineWidth', 4)
xlabel('Online-algoritmissa suoritettujen kierrosten lukumäärä')
ylabel('Kokonaislaskenta-aika (s)')
legend('DY-ratkaisijan kokonaislaskenta-aika', ...
    'LU-hajotelman kokonaislaskenta-aika')
hold off
```